

# 4D Draw®

---

## *Language Reference Windows and Mac OS Versions*



---

## **4D Draw Language Reference**

### **Version 2004 for Windows® and Mac™ OS**

Copyright © 1985-2004 4D SA/4D, Inc.  
All rights reserved

---

The Software described in this manual is governed by the grant of license in the 4D Product Line License Agreement provided with the Software in this package. The Software, this manual, and all documentation included with the Software are copyrighted and may not be reproduced in whole or in part except for in accordance with the 4D Product Line License Agreement.

4D Draw, 4D View, 4D Write, 4th Dimension, 4D, the 4D logo and 4D Server are registered trademarks of 4D SA.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Apple, Macintosh, Mac OS and QuickTime are trademarks or registered trademarks of Apple Computer, Inc.

Mac2Win Software Copyright © 1990-2004, is a product of Altura Software, Inc. This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

4th Dimension includes cryptographic software written by Eric Young (eay@cryptsoft.com)

4th Dimension includes software written by Tim Hudson (tjh@cryptsoft.com).

ACROBAT © Copyright 1987-2004, Secret Commercial Adobe Systems Inc. All rights reserved. ACROBAT is a registered trademark of Adobe Systems Inc.

All other referenced trade names are trademarks or registered trademarks of their respective holders.

# Contents

## 1. 4D Draw, Introduction to the Language... 9

Introduction.....	11
Multi-platform Document Management.....	12
Commands in the Method Editor.....	14
Documents in 4D Draw Areas.....	16
Using the Default Area.....	19
Manipulating Objects.....	20
Using Menu Items.....	23

## 2. DR Area Control..... 25

DR ADD TO BACKGROUND.....	27
DR DO COMMAND.....	28
DR Error.....	29
DR EVENT FILTER.....	30
DR EXPERT COMMAND.....	31
DR EXPERT MODE.....	32
DR GET AREA BOUNDARY.....	33
DR Get update mode.....	35
DR GET MOUSE.....	36
DR Get zoom.....	37
DR LAST CLICK.....	38
DR Last event.....	39
DR MENU STATUS.....	40
DR ON ERROR.....	41
DR ON EVENT.....	42
DR ON MENU.....	44
DR REDRAW.....	45
DR RELEASE BACKGROUND.....	46
DR REMOVE FROM BACKGROUND.....	47
DR SCROLL DOCUMENT.....	48
DR SET ENTERABLE.....	49
DR SET UPDATE MODE.....	50
DR ZOOM.....	51

### 3. DR Area Options.....53

DR COORDINATES.....	55
DR DISPLAY OPTIONS.....	56
DR Get display.....	57
DR GET DOCUMENT SIZE.....	58
DR Get draw mode.....	59
DR GET GLOBAL PREFERENCES.....	60
DR GET PREFERENCES.....	61
DR SET DISPLAY.....	63
DR SET DOCUMENT SIZE.....	65
DR SET DRAW MODE.....	66
DR SET GLOBAL PREFERENCES.....	67
DR SET PREFERENCES.....	69

### 4. DR Areas.....71

DR AREA TO AREA.....	73
DR AREA TO FIELD.....	74
DR Area to picture.....	76
DR DELETE OFFSCREEN AREA.....	77
DR FIELD TO AREA.....	78
DR NEW DRAWING.....	79
DR New offscreen area.....	80
DR OPEN DOCUMENT.....	81
DR PICTURE TO AREA.....	83
DR SAVE DOCUMENT.....	84

### 5. DR Binding.....87

Binding Commands, Introduction.....	89
DR ACTIVATE BIND.....	93
DR ADD TO BIND.....	94
DR DEACTIVATE BIND.....	95
DR DELETE BIND.....	96
DR New bind.....	97
DR REMOVE FROM BIND.....	98

## 6. DR Get Attributes..... 99

DR GET ARC SPECS.....	101
DR Get attribute lock.....	103
DR GET BOUNDARY.....	104
DR Get corner rounding.....	105
DR GET ENDMARKS.....	106
DR GET FILL ATTRIBUTES.....	108
DR Get handle state.....	110
DR GET HIGHLIGHT.....	111
DR Get ID.....	112
DR GET LINE ATTRIBUTES.....	114
DR GET LINE SPECS.....	115
DR Get name.....	117
DR Get object type.....	119
DR GET POLYGON VERTEX.....	121
DR Get refnum.....	122
DR Get rotation.....	123
DR Get text.....	124
DR GET TEXT ATTRIBUTES.....	125
DR Get text width.....	127

## 7. DR Import and Export..... 129

DR ARRAY TO ATTRIBUTE.....	131
DR Array to polygon.....	133
DR ATTRIBUTE TO ARRAY.....	135
DR POLYGON TO ARRAY.....	137

## 8. DR Object Creation..... 139

DR Draw arc.....	141
DR Draw line.....	143
DR Draw oval.....	144
DR Draw rectangle.....	145
DR Draw text.....	147
DR End polygon.....	148

DR Objects to bitmap.....	149
DR PLACE PICTURE.....	150
DR POLYGON CURVE.....	151
DR POLYGON LINE.....	153
DR START POLYGON.....	154

## 9. DR Object Manipulation..... 155

DR ADD TO BITMAP.....	157
DR ALIGN.....	158
DR Count.....	161
DR DELETE.....	162
DR GROUP.....	163
DR HIDE.....	164
DR LOCK.....	165
DR MOVE.....	167
DR ROTATE.....	168
DR SCALE.....	169
DR SIZE.....	171
DR UNGROUP.....	173

## 10. DR Object Selection..... 175

DR SELECT.....	177
DR SELECT BY ATTRIBUTE.....	178
DR SELECT BY REGION.....	180

## 11. DR Printing..... 181

DR PRINT.....	183
DR PRINT BACKGROUND.....	184
DR PRINT FOREGROUND.....	185
DR PRINT MERGE.....	186

## 12. DR References..... 187

DR INSERT EXPRESSION.....	189
DR INSERT FIELD.....	191
DR Place field.....	193
DR SET FORMAT.....	194

## 13. DR Rulers..... 195

DR ARRAY BASE TO SCALE.....	197
DR ARRAY SCALE TO BASE.....	198
DR Base to scale.....	199
DR GET ORIGIN.....	200
DR GET RULER.....	201
DR GET RULER OPTIONS.....	203
DR Scale to base.....	204
DR SET ORIGIN.....	205
DR SET RULER.....	206
DR SET RULER OPTIONS.....	207

## 14. DR Set Attributes..... 209

DR SET ARC SPECS.....	211
DR SET ATTRIBUTE LOCK.....	212
DR SET CORNER ROUNDING.....	213
DR SET ENDMARKS.....	214
DR SET FILL ATTRIBUTES.....	215
DR SET HANDLE STATE.....	216
DR SET HIGHLIGHT.....	217
DR SET LINE ATTRIBUTES.....	219
DR SET LINE SPECS.....	220
DR SET NAME.....	221
DR SET POLYGON VERTEX.....	222
DR SET REFNUM.....	223
DR SET TEXT.....	224
DR SET TEXT ATTRIBUTES.....	225

## **15. DR Utilities.....227**

DR Calculate area.....	229
DR Calculate perimeter.....	230
DR Clipboard to picture.....	231
DR Color to index.....	232
DR COLOR TO RGB.....	233
DR Font name.....	234
DR Font number.....	235
DR Index to color.....	236
DR PICTURE TO CLIPBOARD.....	237
DR RGB to color.....	238

## **16. Appendixes.....239**

Appendix A, Attribute Codes.....	241
Appendix B, Error Codes.....	243
Appendix C, Event Codes.....	246
Appendix D, Object Codes.....	247
Appendix E, Unit Codes.....	248
Appendix F, Command Codes.....	249

## **Command Index.....253**



1

---

# 4D Draw, Introduction to the Language



4D Draw is a plug-in that adds almost 150 commands to the 4th Dimension procedural language. With these commands, you can automate tasks typically done manually on a document, such as:

- Execute 4D Draw menu commands
- Open and save documents
- Draw, select, and modify objects

#### 4D Draw documentation

The documentation available for 4D Draw consists of two manuals, the *4D Draw User Reference* and the *4D Draw Language Reference*. The purpose of this manual (4D Draw Language Reference) is to describe the use of the programming language of 4D Draw. For more information about how to use 4D Draw, please refer to the 4D Draw User Reference manual.

#### Language Conventions

All the 4D Draw commands are preceded by the letters DR to distinguish them from standard 4th Dimension commands and from commands added by the other plug-ins. In this manual, 4D Draw commands are printed in uppercase in a special font: DR LOCK. 4D Draw functions are shown with an initial capital letter: DR Count.

When 4D Draw commands or functions appear in methods, they are displayed in a bold italic typeface to differentiate them from built-in 4th Dimension commands and functions. Non-italic bold text indicates 4th Dimension language terms.

In some examples in this manual, a line of code may be continued on a second or third line due to space limitations. However, when you type these examples, keep those lines of code on a single line—do not press the Return key and cause a break in the flow.

4D Draw, like 4th Dimension and 4D Server, is a multi-platform program. Therefore, a database that uses 4D Draw, created under MacOS, can be opened and run under Windows with no modifications, and vice versa. This is possible only if you are using the correct versions of the software.

## Management Principles

To manage 4D Draw documents on both platforms, you must consider the following principles:

- Under MacOS, 4D Draw uses the file type to recognize its own documents. For example, type 4DRW means that the document is a 4D Draw file. The complete access path includes the disk name, folder names, and document name, each separated by a colon (:). For example,

MyDisk:Folder1:Folder2:Mydatabase

- Under Windows, 4D Draw uses the file name extension to recognize its own documents. For example, the .4DW extension means that it is a 4D Draw document. The complete access path includes the disk letter, folder names, and document name, each separated by a backslash (\). For example,

D:\Folder1\Folder2\Mydatabase

- A 4D Draw document created under MacOS and copied onto Windows can be opened directly, provided that it has been saved with its file name extension. For example, the MyDoc document saved under the name MyDoc.4WR, and copied onto a volume of your PC, can be opened directly from the database under Windows, with no further handling.
- A 4D Draw document created under Windows and copied onto MacOS can be opened directly.

## File Equivalents on MacOS and Windows

The following table lists the file equivalents of 4D Draw documents on MacOS and Windows. For more information, refer to the descriptions of the commands DR OPEN DOCUMENT and DR SAVE DOCUMENT.

Document	MacOS Type	Windows Extension
4D Draw	4DRW	.4DW
MacPaint	PNTG	.PNT
EPSF	EPSF	.EPS
PICT	PICT	.PCT

## Templates

To share a template on both Mac OS and Windows client machines, regardless of whether the server is a machine under MacOS or Windows, the program functions in the same manner.

While importing a template (single user or client/server), 4D Draw tries to read the AreaName\_.4DW file, where .4DW is the Windows file name extension for the document. If the file is not found, for the sake of compatibility, the module will then try to read the file named AreaName\_ file, with no file name extension.

In saving a template, the module will always use the file name AreaName\_.4DW, regardless of the platform.

## Managing Fonts

Font management procedures differ under MacOS and Windows. Under MacOS, each installed font has a different name and number. Fonts are thus applied by referencing these two unique identifiers.

Under Windows, fonts are also referenced by name and number, but the number varies with the font attributes. Each attribute of a font, such as bold, italic, etc., will have a different number. Therefore, different font numbers can return the same font name.

However, a font name always returns the same number, that of the “normal” style of that font. The ASIFONT.MAP file on your PC lists the names of corresponding MacOS and Windows fonts. You can edit this file with a text editor. In 4D Draw, the following functions are affected: DR Font name and DR Font number.

**Note:** Windows does not provide for the Outline and Shadow attributes. Selecting either of these has no effect in the Windows environment.

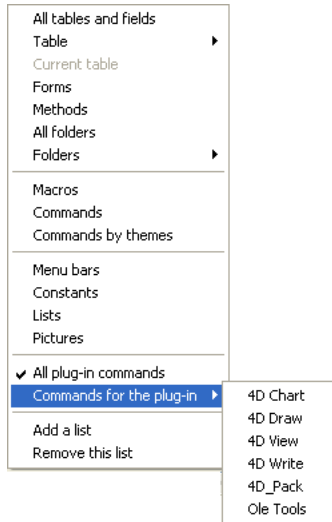
### Application/System Font

The values returned by DR Font name and DR Font number depend on the fonts installed in your system. The following two fonts each have a constant unique number: System font and Application font:

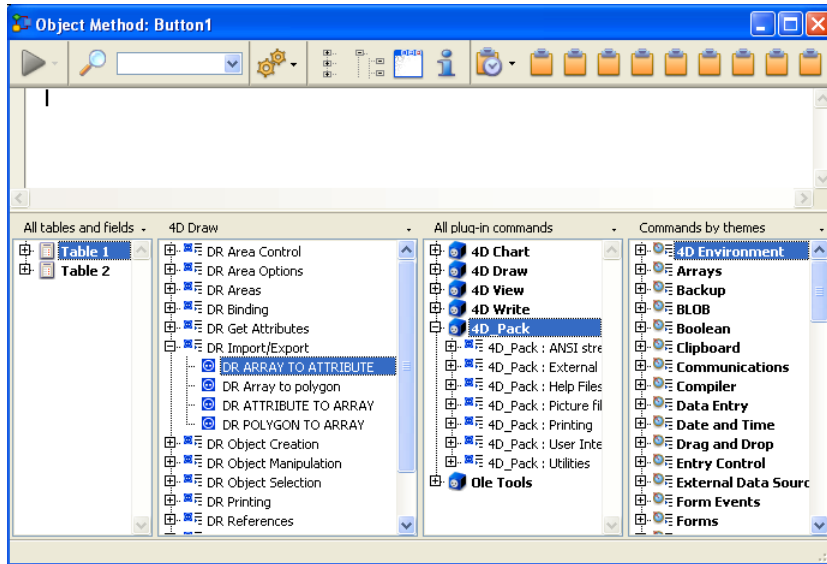
The Application font always has the number 1; however, you must also note the following behavior:

- Under Windows, for 4D Draw, the Application font is Arial. Normally, Arial has the number 21. Therefore, DR Font number(1) returns Arial and DR Font number("Arial") returns 21.
- Under MacOS, for 4D Draw, the Application font is Geneva. Normally, Geneva has the number 3. Therefore, DR Font number(1) returns Geneva and DR Font number("Geneva") returns 3.

4D Draw commands can be displayed in a list in the 4th Dimension Method editor. The list can contain either the 4D Draw commands only, or all the available plug-in commands:



Plug-ins commands are grouped in “themes” in hierarchical lists:



Plug-ins commands are also displayed on the Components page of the Explorer.

You can insert a 4D Draw command in a method just as you do for any 4th Dimension command: you can either type it directly into the Method editor or double-click the command name in the list.

A 4D Draw document can exist in any of three locations (areas) in 4th Dimension:

- External areas in forms
- External windows
- Offscreen areas

To use a 4D Draw document, you either create an external area on a layout or open an external window. You create an external area by drawing the area on a layout in the Design environment. You open an external window either by choosing 4D Draw from the Windows menu in the User environment or by executing the Open external window function.

In addition to creating visible areas, you can create invisible offscreen areas.

### Area ID Numbers and Area Variables

When you use commands to control a 4D Draw document, you need to identify the document by its area ID number. Wherever the document is located, 4D Draw needs its area ID number to locate it and operate on it. The area ID number is internal to 4D Draw and is normally stored in a variable.

4D Draw uses variables to store the location of 4D Draw areas, external windows, and offscreen areas. You reference the area on which you want to perform an operation by passing the variable containing the area's ID number as a parameter to the command or function.

In the command descriptions that follow this introduction, the area parameter refers to the variable identifying the document area. There are two types of area variables:

- External area object names
- Variables you create for an external window or offscreen area

### External area object names

When you create and name a 4D Draw area, 4th Dimension automatically recognizes the name of the 4D Draw area as a variable referring to the area. For example, you would refer to the FloorPlan area by specifying "FloorPlan" as the area parameter.

### Variables for external window or offscreen area

When you create an external window or offscreen area using the Open external window or DR New offscreen area functions, the area ID number returned by the function should be stored in a variable. You can then use the variable to refer to the external window or offscreen area in other commands and functions. To store the value in a variable, you place the variable name and the assignment operator (:=) to the left of the function in the line of code. Most 4D Draw commands require you to specify an area before they can be executed.



- The following example creates a 4D Draw external window and stores the area ID number in the *MyArea* variable:

```
MyArea:=Open external window(30;30;350;450;8;"Design";"_4D Draw")
```

#### 4D Draw External Areas on Forms

When you want a 4D Draw document to appear in a 4th Dimension form, you must create an external area on the form and assign it a unique name, specifying the external area type as 4D Draw. You can place 4D Draw areas in an input form, so that you can work with documents, or in an output form to display and print information.

4D Draw can use the entire form, or it can share space with fields and other form elements.

You use an active object area of the external object type for 4D Draw. An external object is one of several types of active objects in 4th Dimension, such as buttons, enterable areas, and scrollable areas. For complete information about external objects, see the *4th Dimension Design Reference* and *Language Reference*.

When you need to refer to a document on a form, use the object name that you used when you created the 4D Draw external object.

#### 4D Draw External Window Areas

4th Dimension allows you to create a 4D Draw document in an independent area called an external window. External windows for 4D Draw are useful when you want the user to have access to a drawing program at any time.

Issuing the 4th Dimension function, *Open external window*, from a method opens a specified window and returns an area ID in a long integer variable. You can reference this variable whenever you want to issue a 4D Draw command to affect the external window.

For complete information about the *Open external window* function, see the *4th Dimension Language Reference*.

#### 4D Draw Offscreen Areas

An offscreen area is stored in memory and is not visible to the programmer or user. You can use an offscreen area to make modifications to a document before the user views it or to save the document so the user can revert to the original, if necessary.

4D Draw operations in an offscreen area can be performed more quickly because the screen does not have to be redrawn.

You can use the *DR New offscreen area* function to create an offscreen area. You can use the *DR PICTURE TO AREA* command to place a 4D Picture field—which can contain a 4D Draw area—in a 4D Draw area—which can be an offscreen area.

Remember to delete the offscreen area after you are done with it to free the memory it uses. 4th Dimension will display an error message when you close the database if you have not cleared all offscreen areas.

When placed in a project method, the code in the following example creates an offscreen area for saving a document. Using a button on a form, you can allow the user to revert to the original saved document.

```
Area:=DR New offscreen area
QUERY([Designs];[Designs]CustID=vCustID)
If(Records in selection([Designs]=1)
  Area:=DR PICTURE TO AREA(Area;[Designs]FloorPlan_)
  `Store the floor plan in the offscreen area
  MODIFY RECORD([Designs])
  `Modify the designs record
  DR DELETE OFFSCREEN AREA(Area)
  `Free the memory used by the offscreen area
End if
```

Create a button on the input form and assign it the following code:

```
Review:=DR Area to picture(Area;-2)
  `Places the offscreen area that contains the original document
  `into the external area contained in the FloorPlan form.
```

The default area is a template in RAM that can be used to set the default attributes of all 4D Draw areas and external windows. Any command that can be executed on a 4D Draw area can be executed on the default area by setting the area parameter to -1. You can use methods to perform operations on the default area as you would with any other area.

By using the default area, you can eliminate the unnecessary execution of code for 4D Draw areas. For example, if you want all 4D Draw areas and external windows to appear without scroll bars, you don't have to turn the scroll bars off in the On load event of each form.

You can set the attributes of both 4D Draw areas and external windows. The default area is automatically used as a template whenever a form or external window is opened. Since no code has to be executed, the default area provides a quick way to customize the drawing area.

If you do not want the default area to apply to every 4D Draw area, you can override it by creating a template on disk for the 4D Draw area or by placing the appropriate code in the On load form event of the form. A template on disk or code in the On load form event takes precedence over the default area.

### Object ID Numbers

Every object in a 4D Draw document is given a unique number. This number is referred to as the object's ID and is assigned when the object is created.

This means that each time the user draws an object with the palette, pastes an object from the Clipboard, groups several objects, duplicates an existing object, or pastes a field reference, a new ID is assigned. Since the object ID is unique, IDs are a convenient way to refer to objects. Object IDs are never reused within a document. Even if an object is deleted, its number is "retired" for the life of the document.

Object IDs cannot be transferred. An object whose ID is equal to 5 in one 4D Draw document will not necessarily have the same ID if it is pasted into another document. All of the standard object creation commands are functions and return the resulting object's ID. Unlike objects created by the user through the palette, objects created through the commands are not automatically selected.

You can obtain the ID of an object using the DR Get ID function.

### Disabling Access to Text Objects

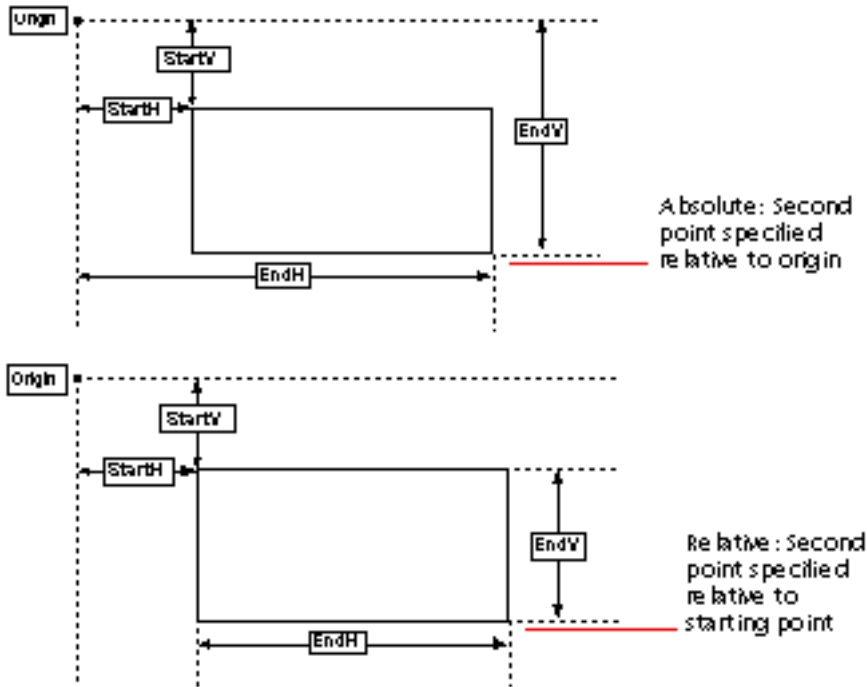
If you want to disable access to text object attributes, you must deactivate both the Text menu and the Attributes menu item in the Object menu. A text object is accessible by either means.

If you do not want to disable access to the Attributes menu item for objects other than text objects, you can install a method using DR ON MENU to test if the menu item has been selected. You need only determine the type of the selection the user made before choosing the menu item. If the selection contains an object of type Text, you can display an alert box. If not, you can execute the menu item using the DR DO COMMAND command.

### Specifying an Object's Coordinates

An object's position and size are referred to as its coordinates. All commands that describe or specify coordinates do so in base units. Base units are the units specified in the Base Units pop-up menu in the Rulers dialog box. Use the DR Scale to base and DR Base to scale functions to convert between base and scale units.

Most commands that describe or specify a position do so with respect to the origin. The origin is the intersection of the zero points on the horizontal and vertical rulers. In some commands, an option is given to specify the ending point with respect to the starting point. For instance, in DR Draw rectangle, the first point (startH/startV) is always specified using the origin, but the second point (endH/endV) can use either the origin or an offset from the starting point. The following diagrams illustrate the difference between absolute and relative measurement:



A positive coordinate indicates a position to the right or below. A negative coordinate indicates a position to the left or above.

## Specifying the Scope of a Command

Many 4D Draw commands have a parameter called *scope*, which specifies those objects or text characters in a 4D Draw document that are affected by the command. The following table describes the general rules for scope. For a description of how scope affects a given command, see the command description.

Scope	Affected text or object
>0	Selected objects
0	Selected objects
-1	All objects in the document
-2	Default values
-3	Selected characters
-4	Foreground objects
-5	Background objects

## Modifying Hidden and Background Objects

Objects that are hidden (invisible) or that are part of the background can be selected only by using the 4D Draw language. When selecting objects that are hidden or on the background, we strongly recommend that you perform the following steps before returning control to the user.

1. Select the objects.
2. Perform the action on the objects.
3. Deselect the objects.

This last step is important because the user may inadvertently perform an action on the objects if they remain selected.

You can procedurally gain access to a 4D Draw menu and select a menu item (command). In a procedure, you can determine the status of a menu or menu item. Each menu item is referenced by a unique integer.

Menu item integers are generally based on the location of the menu and menu item. The menus are numbered from left to right in ascending order. For example, File = 1000 and Edit = 2000. Likewise, menu items are numbered in ascending order from top to bottom. Therefore, the New menu item is numbered 1001, because it is the first item on the first menu, File.

The numbers of these menu items always remain the same, even in future versions in which new menu items may be added. Any new menu items will use different numbers, even if they are placed between current menu items. This placement will invalidate the general rule of numbering menu items, but the menu references used in methods will remain accurate, so you will not need to update them.

See Appendix F, Command Codes for a listing of menu item code integers as well as codes used by the 4D Draw Tool Palette.





**2**

---

# **DR Area Control**



DR ADD TO BACKGROUND (area; scope)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All objects 0=Selected objects >0=ID

### Description

The command DR ADD TO BACKGROUND adds the objects specified by scope to the background of area. This command has the same function as the **Add to Background** menu item.

The objects specified by scope become part of area's background and are no longer selectable. To modify the objects, you must release the background using either the **Release Background** menu item or the DR RELEASE BACKGROUND and DR REMOVE FROM BACKGROUND commands. Background objects can be modified only procedurally.

### See Also

DR REMOVE FROM BACKGROUND.

DR DO COMMAND (area; command)

Parameter	Type		Description
area	Longint	→	4D Draw area
command	Longint	→	Number of command

### Description

The command DR DO COMMAND executes the menu item specified by command. The menu item is executed as if the user had chosen it from a 4D Draw menu. Use this command to perform any actions that do not have a procedural equivalent.

The possible values for command are listed in Appendix F, Command Codes. These numbers will remain the same even if menu items are changed or moved in future versions of 4D Draw.

### Example

The following example sends the selected objects in Area to the back of the drawing.

⇒ *DR DO COMMAND* (Area;5002)

### See Also

DR MENU STATUS.

---

DR Error {(message)} → Integer

Parameter	Type		Description
message	Text	←	Receives error message
Function result	Integer	←	Status of the last operation performed by 4D Draw

### Description

The command DR Error returns a number that represents the status of the last operation performed by 4D Draw.

If DR Error equals 0, the last operation did not cause an error. If DR Error does not equal 0, an error occurred during the last operation. If several areas are active on the same form, DR Error returns the last error, without discriminating between areas. See Appendix C, Event Codes for a complete list of event codes.

If the optional message parameter is passed to DR Error, it must be a variable and will contain the text of the error after the call.

DR Error could return a 4D error code if the error is a general database management error.

### See Also

DR ON ERROR.

DR EVENT FILTER (area; mask)

Parameter	Type		Description
area	Longint	→	4D Draw area
mask	Longint	→	Events to process

### Description

The command DR EVENT FILTER specifies which events cause the object method of area or the event method to be executed. By default, an object method attached to a 4D Draw area is executed when the user selects an object outside of the area. DR EVENT FILTER lets you specify other events that execute the object method. Also, a method installed with the DR ON EVENT command will also execute.

mask is the events to use, expressed as the sum of event codes. See Appendix C, Event Codes, for a complete list of event codes.

If you specify -1 for area, the event filter becomes the default filter for all newly created 4D Draw areas on forms and in external windows. This lets you trap for areas created from the Plug-Ins menu of the User environment. It also allows for consistent event handling in all areas.

### See Also

DR LAST CLICK, DR Last event, DR ON EVENT.

---

DR EXPERT COMMAND (area; command; status)

Parameter	Type		Description
area	Longint	→	4D Draw area
command	Longint	→	Number of command
status	Integer	→	0=Enabled 1=Disabled

### Description

The command DR EXPERT COMMAND enables or disables a menu command for 4D Draw expert mode.

If status equals 0, the menu command specified by command is enabled in expert mode. If status equals 1, the item is disabled. The possible values for command are listed in Appendix F, Command Codes.

If a menu item is disabled with DR EXPERT COMMAND, it can still be executed by calling DR DO COMMAND.

Disabling certain menu items affects other operations in 4D Draw:

- If you disable the **Goto Full Window** item, the **Zoom** button is also removed. If you disable the **Attributes** menu item, the user cannot double-click an object to display the **Object Attributes** dialog box.
- If you disable the **Locked** menu item, the **Lock** button in the **Object Attributes** dialog box is also disabled.

### Example

The following example disables the **Attributes** menu item and stops the user from opening the **Object Attributes** dialog box:

```
⇒ DR EXPERT COMMAND (Area;4018;1)
   DR EXPERT MODE (Area;1)
```

### See Also

DR EXPERT MODE.

---

DR EXPERT MODE (area; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
mode	Integer	→	1=Expert mode On, 0=Expert mode Off, 666=Print one job mode On, 667=Print one job mode Off

### Description

The DR EXPERT MODE command allows turning special modes on or off in the 4D Draw area for the current session: expert mode and "print one job" mode.

### Expert Mode

When 4D Draw is in expert mode, certain items on 4D Draw menus may be disabled. When expert mode is invoked, the menu items previously specified with DR EXPERT COMMAND are disabled.

- If mode equals 1, expert mode is invoked.
- If mode equals 0, expert mode is off (standard).

### "Print one job" Mode

When this mode is enabled, 4D Draw stores all the print requests in a temporary file on disk. The documents will only be sent to the printer when "print one job" mode has been disabled (using value 667). This lets you make sure that all the documents will be printed in a single print job. This is especially useful when printing PDF documents.

- To enable "print one job" mode, pass 666 in the mode parameter,
- To disable this mode (default behavior), pass 667 in the mode parameter.

### Examples

(1) See the example for DR EXPERT COMMAND.

(2) Here is a typical example for enabling "print one job" mode:

```

$Area:=DR New offscreen area
⇒ DR EXPORT MODE($Area;666)
  ALL RECORDS([MyTable3])
  For($i;1;Records in selection([MyTable3]))
    DR AREA TO FIELD($Area;3;4)
    DR PRINT($Area;0)
    NEXT RECORD([MyTable3])
  End for
⇒ DR EXPORT MODE($Area;667)
  DR DELETE OFFSCREEN AREA($Area)

```

### See Also

DR EXPERT COMMAND.



---

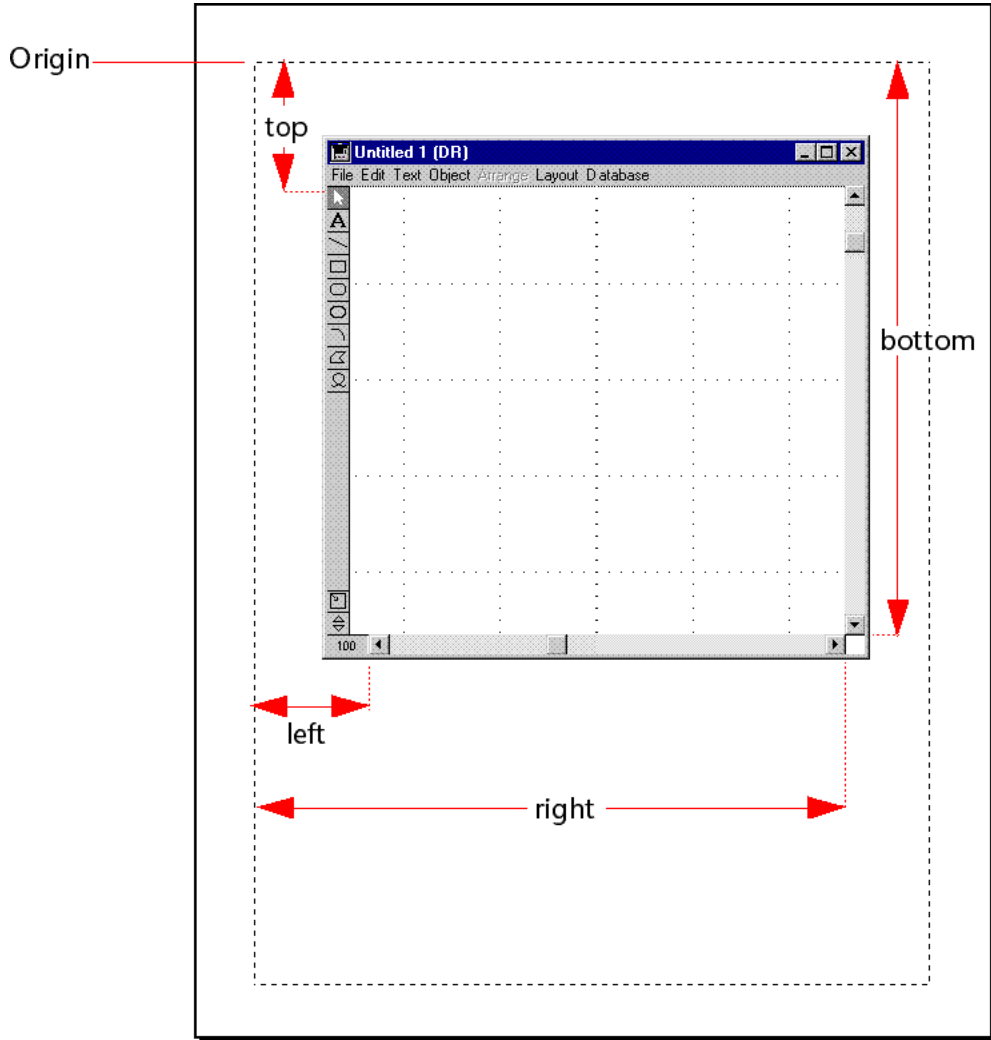
DR GET AREA BOUNDARY (area; left; top; right; bottom)

Parameter	Type		Description
area	Longint	→	4D Draw area
left	Number	←	Left boundary of area
top	Number	←	Top boundary of area
right	Number	←	Right boundary of area
bottom	Number	←	Bottom boundary of area

### Description

The command DR GET AREA BOUNDARY returns into the left, top, right, and bottom variables, the coordinates for the rectangular viewing region of area. The coordinates are expressed in base units according to the current origin. Use the DR Base to scale function to convert from base units to scale units.

The following diagram explains the left, top, right, and bottom parameters:



### Example

The following example moves the selected objects to the upper-left corner of the viewable area.

⇒ *DR GET AREA BOUNDARY* (Area;\$Left;\$Top;\$Right;\$Bottom)  
*DR MOVE* (Area;0;\$Left;\$Top;0)

See Also

DR GET BOUNDARY.

---

DR Get update mode (area) → Integer

Parameter	Type		Description
area	Longint	→	4D Draw area
Function result	Integer	←	Screen update status

### Description

The command DR Get update mode returns an integer that describes the screen update status for area. It is useful for designing modular programs.

If DR Get update mode equals 0, screen updating is turned off. If DR Get update mode equals 1, screen updating is turned on.

### Example

The following example turns off screen updating, calls a project method that makes several modifications, and then resets the screen update mode. This is particularly useful when working with nested methods.

```
⇒ $Mode := DR Get update mode (Area)
   `Get the current update mode
   DR SET UPDATE MODE (Area;0)
   `Turn off screen updating
   REFORMAT (Area)
   `Area is a Longint
   DR SET UPDATE MODE (Area;$Mode)
   `Restore original update mode
```

### See Also

DR REDRAW, DR SET UPDATE MODE.

---

DR GET MOUSE (area; horizontal; vertical)

Parameter	Type		Description
area	Longint	→	4D Draw area
horizontal	Number	←	Horizontal position of mouse cursor
vertical	Number	←	Vertical position of mouse cursor

### Description

The DR GET MOUSE command returns the current location of the mouse cursor in the horizontal and vertical variables. This position is calculated with respect to the origin of area and is expressed in base units (inches, points, centimeters). The DR Base to scale command can be used to convert the base units to scale units.

The command returns the location of the mouse cursor, regardless of whether or not the mouse button is pressed. During a dragging or resizing operation, DR GET MOUSE returns the position of the mouse when its button is released.

If the command has been correctly executed, the system variable OK is set to 1; otherwise, it is set to 0.

### Example

This example describes the installation of a process method displaying the coordinates of the mouse in a separate window when it moves over the 4D Draw area.

- The *zDrInfoGetMouse* method launches the process which returns the mouse position:

```

`zDrInfoGetMouse method
C_LONGINT($NewProc)
$NewProc:=New process("zDrGetMouse";256;"MyTest";Drawarea)

```

- The *zDrGetMouse* method, called using the \$NewProc process, displays the X and Y coordinates of the mouse in a window until the user presses the Shift key:

```

`zDrGetMouse method
C_REAL($MouseX;$MouseY)
$ref:=Open window(50;80;300;125;-Palette window;"Mouse position";"CloseBox")
GOTO XY(1;0)
MESSAGE("Press Shift to exit process")
GOTO XY(1;1)
MESSAGE("Values in X / Y")
Repeat
⇒ DR GET MOUSE ($1;$MouseX;$MouseY)
   GOTO XY(1;2)
   MESSAGE(String($MouseX)+" / "+String($MouseY)+"")

```

DR Get zoom (area) → Number

Parameter	Type		Description
area	Longint	→	4D Draw area
Function result	Number	←	Zoom percentage for the 4D Draw area

**Description**

The command DR Get zoom returns the zoom percentage for the 4D Draw area area. The zoom percentage returned by this command is the number displayed in the Zoom Percentage indicator in the lower left corner of the 4D Draw area.

---

DR LAST CLICK (area; horizontal; vertical)

Parameter	Type		Description
area	Longint	→	4D Draw area
horizontal	Number	←	Horizontal position of the last click
vertical	Number	←	Vertical position of the last click

### Description

The command DR LAST CLICK returns into the horizontal and vertical variables the position of the last mouse click in area according to the current position of the origin.

Both horizontal and vertical are expressed in base units. Use the DR Base to scale function to convert from base units to scale units.

DR LAST CLICK returns the point where the mouse was clicked, not where it was released. For an action such as dragging or resizing, DR LAST CLICK returns the position that the pointer was in when the mouse button was pressed. You can use DR LAST CLICK in the object method of an area or in an event method to obtain the position where an object was clicked.

### Example

The following example is the object method for a 4D Draw area on a form. Each time the object method is executed, DR LAST CLICK places the mouse position into the *vHorizontal* and *vVertical* variables. If these variables are displayed on the form the user will have immediate feedback on exactly where the mouse was clicked.

⇒ *DR LAST CLICK* (Area;vHorizontal;vVertical)

### See Also

DR EVENT FILTER, DR Last event.

---

DR Last event (area) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
Function result	Longint	←	Code of the last event that occurred in area

### Description

The command DR Last event returns the code of the last event that occurred in area.

DR Last event can be used in the object method of a 4D Draw area or in an event method installed with DR ON EVENT. DR Last event identifies the event that caused the object or project method to execute. When used with the DR EVENT FILTER command, DR Last event allows you to take action based on a user's actions. See Appendix C, Event Codes, for a complete list of event codes.

### Example

The following example is the object method for a 4D Draw area. It first tests to see if the object method ran because of a Command-click (or Ctrl-click in Windows) and then tests to see if only one object is selected. If both criteria are met, it passes the object's reference number to a project method.

```
⇒  If (DR Last event (Area) = 64)
    `Was there a Command-Click
    If (DR Count (Area;0) = 1)
        `How many objects are selected
        PART INFO (DR GET ID(Area;0;1))
        `Pass ref number to PART INFO.
    End if
End if
```

The following is the *PART INFO* method. This method searches for the part, calls a project method that opens a centered window, and then displays the record for modification.

```
QUERY ([Parts];[Parts]ID=$1) `Search for record that goes with object
CENTER WINDOW (400;300) `Open a centered window
MODIFY RECORD ([Parts];*) `Allow the user to make modifications
CLOSE WINDOW `Close the window
```

### See Also

DR EVENT FILTER, DR ON ERROR.

---

DR MENU STATUS (area; command; checked; active; name)

Parameter	Type		Description
area	Longint	→	4D Draw area
command	Longint	→	Number of command
checked	Integer	←	0=Not checked 1=Checked
active	Integer	←	0=Inactive 1=Active
name	String	←	Name of menu item

### Description

The command DR MENU STATUS returns in the checked, active, and name variables information about the menu item in area represented by command.

The possible values for command are listed in Appendix F, Command Codes.

- If active equals 0, the menu item is disabled. If active equals 1, the menu item is enabled.
- If checked equals 0, the menu item is not checked. If checked equals 1, the menu item is checked. The menu items for which checked is meaningful are Reshape, Rotate, Actual Size, Fit to Window, Snap To Grid, and all items in the Display, Font, Size, Style, and Justification submenus. Checked also is meaningful for Show Values/Show References, even though neither has a check mark. If Show Values is displayed on the menu, then the document is currently displaying references in text objects and checked contains 0. If Show References is displayed on the menu, then the document is currently displaying values in text objects and checked contains 1.
- name is the text of the menu item.

### Example

The following example uses DR MENU STATUS to see if the grid is currently on.

```
⇒ DR MENU STATUS (Area;6008;vChecked;vActive;vName) `6008 is "Snap to Grid"
   If (vChecked=1) `If grid is on
     vMessage := "The grid is on." `Let the user know
   End if
```

### See Also

DR DO COMMAND.



---

DR ON ERROR (method)

Parameter	Type		Description
method	Text	→	Method to execute

### Description

The command DR ON ERROR installs method to manage 4D Draw errors.

If method is an empty string, no method is called and error handling returns to 4D Draw. After installation, 4D Draw calls method when a 4D Draw error occurs.

When 4D Draw calls method, it returns three parameters (\$1, \$2, and \$3) that can be used to manage the error:

- \$1 is a long integer that represents the 4D Draw area where the error took place. If the error is not specific to a 4D Draw area, \$1 equals 0.
- \$2 is an integer that holds error number.
- \$3 is of type text and contains the text of the error.

\$2 and \$3 are the equivalent to a call to DR Error.

You should declare the types of these parameters if you plan to compile your database, as follows:

```
C_LONGINT ($1;$2)
C_TEXT ($3)
```

### Example

This example shows the installation of an error-handling method:

```
⇒ DR ON ERROR ("DRAW ERROR")
```

The following method is *DRAW ERROR*, which tests \$1 to determine whether the error occurred in Area and then presents an alert with the error number and message:

```
C_LONGINT ($1;$2)
C_TEXT ($3)

If ($1 = Area)
  ALERT ("An error occurred in the 4D Draw area 'Area'")
End if
ALERT ("Error number " + String($2) + Char(13) + $3)
```

### See Also

DR Error.

---

DR ON EVENT (method)

Parameter	Type		Description
method	Text	→	Method to execute

### Description

The command DR ON EVENT executes method whenever a previously specified event occurs.

The events that cause method to execute are described by the DR EVENT FILTER command. If method is an empty string, no method is executed. If the area in which the event occurs has both an object method and an event method, the object method executes last. DR ON EVENT is especially useful for 4D Draw areas in external windows because they cannot have object methods.

When 4D Draw calls method, it returns four parameters (\$1, \$2, \$3, and \$4) that can be used to manage the event.

- \$1 is a long integer that represents the 4D Draw area where the event took place.
- \$2 is an integer that holds the event code. \$2 is the equivalent of a call to DR Last event.
- \$3 is the table number of the form on which the area resides. If \$3 equals -1, the area is in an external window.
- \$4 is the number of the field into which the area is being autosaved. If \$4 equals 0, the area is not being autosaved.

See Appendix C, Event Codes, for a complete list of event codes.

You should declare the types of these parameters in the event method if you plan to compile your database, as follows:

```
C_LONGINT ($1;$2;$3;$4)
```

### Example

The following example shows the installation of an event method. It opens an external window, specifies Command-click (Ctrl-click on Windows) as the event and then installs the event method, *EventProc*.

```

    `Open external window
vArea := Open external window (20;50;400;350;0;"Draw";"_4D DRAW")
⇒ DR ON EVENT ("EventProc")
    `Install the method EventProc
DR EVENT FILTER (vArea;64)
    `Command-Click will call the method

```

The following method is *EventProc*. It checks to see how many objects are selected, and if there is only one, displays the object's current name in a request dialog box to allow the user to change it.

```
C_LONGINT ($1;$2;$3;$4)
If (DR Count (vArea) # 1)
    `More or less than one object selected
    ALERT ("Select only one object!")
    `Alert the user
Else
    $Name := DR Get name (vArea;0)
    `Get the name of the selected object
    $Name := Request ("Object name...";$Name)
    `Allow user to specify new name
    If (OK = 1)
        `If the user accepted the request
        DR SET NAME (vArea;0;$Name)
        `Set the new name
    End if
End if
```

#### See Also

DR EVENT FILTER, DR Last event.

---

DR ON MENU (area; method)

Parameter	Type		Description
area	Longint	→	4D Draw area (-1=all areas)
method	String	→	Name of the method to call

### Description

The command DR ON MENU executes method each time a menu item is activated, in either the User or Runtime environment. The menu item can also be called by using the DR DO COMMAND command, as long as the menu item is called in method.

The called method returns three parameters:

\$1	A Long integer containing the ID for the 4D Draw area
\$2	A Long integer containing the menu item number
\$3	A Long integer containing the number of the modifier key pressed

The \$3 parameter corresponds to one of the following modifier keys (or combination of modifier keys):

	MacOS	Windows
0	No modifier	No modifier
1	Command key	Ctrl key
2	Shift key	Shift key
4	Option key	Alt key
8	Control key	

If a combination of modifier keys is pressed, the values are added together and passed as a parameter. For example, a value of 10 indicates that you pressed the Shift and Control keys while selecting a menu item.

**Note:** If you compile your database, you must declare the \$1, \$2, and \$3 parameters as Long integers in the method executed by DR ON MENU.

---

DR REDRAW (area)

Parameter	Type		Description
area	Longint	→	4D Draw area

### Description

The command *DR REDRAW* causes area to be redrawn. This command is useful when you have turned off screen updating with *DR SET UPDATE MODE* and want to redraw a 4D Draw area.

### Example

The following example turns off screen updates, calls a project method that makes several modifications to Area, and then redraws Area without turning screen updating back on.

```

    DR SET UPDATE MODE (Area;0)
      `Turn off screen updating
    REFORMAT (Area)
      `Area is a Longint
⇒ DR REDRAW (Area)
      `Refresh to display changes

```

### See Also

DR Get update mode, *DR SET UPDATE MODE*.

## DR RELEASE BACKGROUND (area)

Parameter	Type		Description
area	Longint	→	4D Draw area

### Description

The command DR RELEASE BACKGROUND releases all of the objects on the background of area and places them in their original positions.

This command has the same function as the Release Background menu item.

### See Also

DR ADD TO BACKGROUND.

DR REMOVE FROM BACKGROUND (area; objectID)

Parameter	Type		Description
area	Longint	→	4D Draw area
objectID	Integer	→	ID number of the object

### Description

The command DR REMOVE FROM BACKGROUND releases the specified object from the background of area and places it in its original position.

### See Also

DR ADD TO BACKGROUND.

---

DR SCROLL DOCUMENT (area; horizontal; vertical; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
horizontal	Number	→	Horizontal position
vertical	Number	→	Vertical position
mode	Integer	→	1=Relative 0=Absolute

### Description

The command **DR SCROLL DOCUMENT** scrolls the document in area according to the horizontal and vertical parameters. Both horizontal and vertical are specified in base units. Use the DR Scale to base function to convert from scale units to base units.

If mode equals 1, area is scrolled to a position horizontal and vertical base units from its current position. If horizontal and vertical are positive numbers, area is scrolled down and to the right. If horizontal and vertical are negative numbers area is scrolled up and to the left.

If mode equals 0, area is scrolled to an absolute position horizontal and vertical base units from the origin.

### Example

The following examples are the object methods for two buttons. One button scrolls the document one base unit to the right and one button scrolls the document one base unit to the left.

```

⇒    `bRight
      DR SCROLL DOCUMENT (Area;1;0;1)
      `bLeft
⇒    DR SCROLL DOCUMENT (Area;-1;0;1)

```

### See Also

DR SET ORIGIN.



---

DR SET ENTERABLE (area; mode{; buttonMode})

Parameter	Type	Description
area	Longint	→ 4D Draw area
mode	Integer	→ 0=Non-enterable 1=Enterable
buttonMode	Integer	→ Display as button when area is small: 0 = Yes, 1 = No

### Description

The DR SET ENTERABLE command controls access to the drawing in area as well as (optionally) the display of area when its size is small.

- If mode equals 1, area is enabled and functions normally. If mode equals 0, area is disabled.

A disabled area cannot be modified by the user, but can be modified by the language. When an area is disabled, the user can scroll through the area and copy the selected objects to the Clipboard. The user cannot change the selection or use the 4D Draw menu bar or the Tool palette.

- The optional buttonMode parameter is used to specify whether the 4D Draw area can be displayed as a button when its size is too small in the form.
  - If you pass 0 in buttonMode, the area is displayed as a button if its height is less than 150 points or if its width is less than 300 points. When the user clicks on this button, 4D Draw switches to full page view. This function is enabled by default.
  - If you pass 1 in buttonMode, the area is never displayed as a button.

### Example

The following example is a form method. It turns off all of the display options in Area and then makes Area non-enterable.

```

If (Form event=On load)
  `If we are in the Before phase
  DR DISPLAY OPTIONS (Area;-1;0)
  `Turn off all display options
⇒ DR SET ENTERABLE (Area;0)
  `Make the area non-enterable
End if

```

### See Also

DR DISPLAY OPTIONS.

---

DR SET UPDATE MODE (area; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
mode	Integer	→	0=Off 1=On

### Description

The command DR SET UPDATE MODE turns screen updating on or off in area.

If mode equals 0, screen updating is turned off. If mode equals 1, screen updating is turned on. This command affects updates caused by both 4D Draw commands and user actions.

When screen updating is off, 4D Draw commands execute faster. For instance, if you need to execute a series of modifications to a 4D Draw area, turn off updating before beginning and then turn updating back on when you are finished. Not only will the commands execute faster, screen refresh will appear much smoother to the user.

Screen updating should always be turned on before the user has access to an area. If not, 4D Draw will not be refreshed when the user clicks or drags in the area.

### Example

The following example turns off screen updating, calls a project method that makes several modifications, and then turns screen updating back on.

```
⇒  DR SET UPDATE MODE (Area;0)
    `Turn off screen updating
    REFORMAT (Area)
    `Area is a Longint
⇒  DR SET UPDATE MODE (Area;1)
    `Turn on screen updating
```

### See Also

DR Get update mode, DR REDRAW.

---

DR ZOOM (area; zoom; horizontal; vertical; placement)

Parameter	Type		Description
area	Longint	→	4D Draw area
zoom	Number	→	Zoom percentage
horizontal	Number	→	Horizontal coordinates
vertical	Number	→	Vertical coordinates
placement	Integer	→	0=Centered 1=Top and left aligned

### Description

The command DR ZOOM allows you to enlarge and reduce the 4D Draw area to zoom percent.

- If placement equals 0, the point at (horizontal, vertical) is centered in the visible area of the 4D Draw area.
- If placement equals 1, the point at (horizontal, vertical) is placed in the top left corner of the area.



**3**

---

# **DR Area Options**



---

DR COORDINATES (area; panellItems; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
panellItems	Longint	→	Attribute to display
mode	Integer	→	0=Hide 1=Show 2=Toggle

### Description

The command DR COORDINATES controls which items are shown when the Coordinates panel is displayed in area. DR COORDINATES is the procedural equivalent of the coordinate check boxes in the Preferences dialog box.

The panellItems parameter specifies the object attributes to affect. panellItems is expressed as the sum of attribute codes. Each code describes one attribute, such as height or width.

The possible values for panellItems are given in the following table:

Code	Coordinate Panel Item
1	Width
2	Height
4	Horizontal Scale
8	Vertical Scale
16	Horizontal Change
32	Vertical Change
64	Line Length
128	Angle

mode controls the display of the coordinates.

- If mode equals 0, the coordinates are hidden.
- If mode equals 1, the coordinates are displayed.
- If mode equals 2, the coordinates are toggled.

---

DR DISPLAY OPTIONS (area; options; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
options	Integer	→	Code for display options
mode	Integer	→	0=Hide 1=Show 2=Toggle

### Description

The command DR DISPLAY OPTIONS controls which display options to show in area. Display options are drawing aids such as ruler lines and page breaks.

options specifies the options to affect. options is expressed as the sum of display option codes. Each code describes one display option. The display option codes are given in the following table:

Code	Option
-1	All options
1	Rulers
2	Ruler lines
4	Page breaks
8	Coordinates
16	Menu bar
32	Tool palette
64	Scroll bar
128	Area border
256	Print area
512	Paper area
1024	Paste board
2048	Zoom box

mode describes how to change the display options:

- If mode equals 0, the options are hidden.
- If mode equals 1, the options are displayed.
- If mode equals 2, the options are toggled.

### Example

The following example turns off all of the display options for Area in the On load event of the form method.

```

If (Form event=On load)
⇒   DR DISPLAY OPTIONS (Area;-1;0)
End if

```



DR Get display (area; optionNum) → Integer

Parameter	Type		Description
area	Longint	→	4D Draw area
optionNum	Integer	→	Option number
Function result	Integer	←	0 = Hidden, 1 = Displayed

### Description

The command DR Get display returns whether the display option optionNum is displayed in area or not.

- If it is displayed, DR Get display returns 1.
- If it is hidden, DR Get display returns 0.

DR GET DOCUMENT SIZE (area; width; height)

Parameter	Type		Description
area	Longint	→	4D Draw area
width	Number	←	Width of document
height	Number	←	Height of document

### Description

The command DR GET DOCUMENT SIZE returns into the width and height variables the size, in base units, of the document in area. Use DR Base to scale to convert from base units to scale units.

### Example

See the example for the DR SET DOCUMENT SIZE command.

### See Also

DR SET DOCUMENT SIZE.

---

DR Get draw mode (area) → Integer

Parameter	Type		Description
area	Longint	→	4D Draw area
Function result	Integer	←	Current drawing mode for area

### Description

The command DR Get draw mode returns a value that describes the current drawing mode for area.

If DR Get draw mode equals 0, the drawing mode is set to draw new objects from their edges.

If DR Get draw mode equals 1, the drawing mode is set to draw new objects from their centers.

### See Also

DR SET DRAW MODE.

---

DR GET GLOBAL PREFERENCES (write; read{; selection{; grid{; horLock{; vertLock{}}))

Parameter	Type		Description
write	Integer	←	Location to which to write 0=Client, 1=Server
read	Integer	←	Location from which to read 0=Client, 1=Server
selection	Integer	←	Object selection 1=Included objects, 2=Touched objects
grid	Integer	←	Default activation of grid 1=Grid disabled, 2=Grid enabled
horLock	String	←	Horizontal lock key
vertLock	String	←	Vertical lock key

### Description

The DR GET GLOBAL PREFERENCES command allows getting the current value for several global options that are used in all 4D Draw areas.

For more information about these options and their values, refer to the description of the DR SET GLOBAL PREFERENCES command.

### See Also

DR SET GLOBAL PREFERENCES.

---

DR GET PREFERENCES (area; order; pict; lockAlerts; autoScroll; variable; confirm; saveMethod)

Parameter	Type		Description
area	Longint	→	4D Draw area
order	Integer	←	Print direction
pict	Integer	→	0=Picture 1=Objects
lockAlerts	Integer	←	0=Off 1=On
autoScroll	Integer	←	0=Off 1=On
variable	Integer	←	0=Fixed 1=Variable
confirm	Integer	←	0=Without confirmation 1=With confirmation
saveMethod	Integer	←	0=Picture and data 1=Picture only 2=Data only

### Description

The command DR GET PREFERENCES returns in the order, pict, lockAlerts, autoScroll, variable, confirm, and saveMethod variables, several parameters for area. These are the features in the Preferences dialog box.

order sets the order in which pages in a multipage document print when area is printed.

- If order equals 0, pages print from top to bottom, then left to right.
- If order equals 1, pages print from left to right, then top to bottom.

The default is to print pages left to right.

picts determines how PICT objects and documents are interpreted in area.

- If pict equals 0, opened, imported, or pasted Picts are interpreted as a single object.
- If pict equals 1, 4D Draw attempts to separate the PICT into its component objects.

The default option is to paste a PICT drawing as a single picture.

lockAlerts determines whether area displays an alert when a user tries to alter a locked attribute.

- If lockAlerts equals 0, no alert is presented.
- If lockAlerts equals 1, the alert is displayed.

The default is to display the alert.

autoScroll determines whether the document in area scrolls automatically when the user drags past the area boundary.

- If autoScroll equals 0, auto scrolling is turned off.
- If autoScroll equals 1, auto scrolling is turned on.

The default is to use auto-scrolling.

variable determines whether area prints using the print variable frame option.

- If variable equals 0, area prints the same size as it was defined and only those objects in the upper left corner of the document are included.
- If variable equals 1, area expands vertically to print all objects on the left side of the document.

The default is to print with a variable frame.

The confirm parameter determines if a confirmation dialog box appears when you accept a record that contains a non-autosaved 4D Draw area. By default, a confirmation dialog box appears, asking whether or not you would like to save the document in the 4D Draw area.

- If confirm equals 0, the confirmation dialog box does not appear.
- If confirm equals 1, the confirmation dialog box appears.

The optional saveMethod parameter controls how the document in the 4D Draw area is saved.

- If saveMethod equals 0, the default value, both the picture and the internal data used to rebuild the image are saved.
- If saveMethod equals 1, only the picture (PICT) is saved. Objects can no longer be manipulated individually.
- If saveMethod equals 2, only the data concerning the objects in the 4D Draw area is saved. The image is later rebuilt using the information in the saved data. This save option is the quickest and uses the least amount of memory. If there is not enough memory for the save method chosen, a dialog box that allows you to choose another method appears.

See Also

DR SET PREFERENCES.

---

DR SET DISPLAY (area; optionNum; display)

Parameter	Type		Description
area	Longint	→	4D Draw area
optionNum	Integer	→	Option number
display	Integer	→	0=Hide 1=Display 2=Toggle

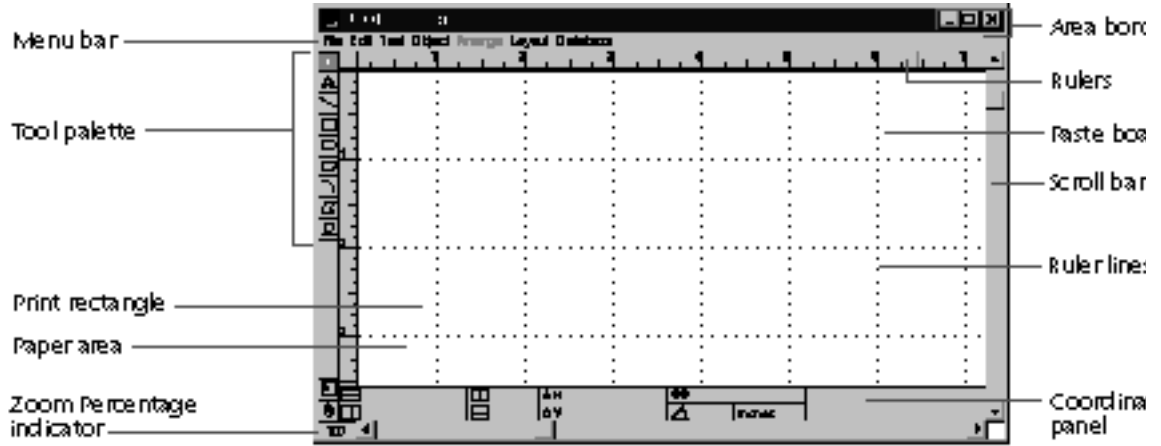
### Description

The command DR SET DISPLAY allows you to manage the display options for the 4D Draw area area, such as whether or not rulers, tools, and other items, are displayed. You can set some of these options using the DR DISPLAY OPTIONS command.

Options for DR SET DISPLAY and DR Get display are as follows:

Number	Option
1	Rulers
2	Ruler lines
3	Page breaks
4	Coordinates panel
5	Menu bar
6	Tool palette
7	Scroll bars
8	Area border
9	Print rectangle
10	Paper area
11	Paste board
12	Zoom Percentage indicator

The following diagram illustrates the options from the table:





---

DR SET DOCUMENT SIZE (area; width; Height)

Parameter	Type		Description
area	Longint	→	4D Draw area
width	Number	→	Width of document
Height	Number	→	Height of document

### Description

The command DR SET DOCUMENT SIZE sets the size of the document in area.

After a call to DR SET DOCUMENT SIZE, the document is width wide and height tall. width and height are expressed in base units. Use the DR Scale to base function to convert from scale units to base units.

### Example

The following example is a project method that is called from the object methods of two enterable areas on a form that contains Area. The two variables are named *vWidth* and *vHeight* and have their Only if Modified check box unchecked. In the Before phase, the method places the current document size into the two variables. In the During phase the method sets the document size to the values typed into the variables.

```

Case of
  : (Before)
⇒      DR SET DOCUMENT SIZE (Area;vWidth;vHeight)
  : (During)
⇒      DR GET DOCUMENT SIZE (Area;vWidth;vHeight)
End case

```

### See Also

DR GET DOCUMENT SIZE.

---

DR SET DRAW MODE (area; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
mode	Integer	→	0=Edge 1=Center

### Description

The command DR SET DRAW MODE controls whether new objects in area are drawn from their centers or from their edges.

If mode equals 0, objects are drawn from their edges.

If mode equals 1, objects are drawn from their centers.

The default setting is to draw objects from their edges.

### Example

The following example is the object method for a check box named bMode. In the Before phase, it determines the drawing mode of Area and sets the check box accordingly. It then allows the user to set the drawing mode by clicking the check box. If the check box is checked, objects in Area are drawn from their centers. If the check box is unchecked, objects are drawn from their edges.

```

Case of
  :(Before)
    `If we are in the Before phase
      bMode := DR Get draw mode (Area)
    `Set bMode equal to the drawing mode
  :(During)
    `If we are in the During phase
⇒    DR SET DRAW MODE (Area;bMode)
    `Set the drawing mode
End case

```

### See Also

DR Get draw mode.

---

DR SET GLOBAL PREFERENCES (write; read{; selection{; grid{; horLock{; vertLock}}))

Parameter	Type		Description
write	Integer	→	Location to which to write -1=No change, 0=Client, 1=Server
read	Integer	→	Location from which to read -1=No change, 0=Client, 1=Server
selection	Integer	→	Object selection 0=No change, 1=Included objects, 2=Touched objects
grid	Integer	→	Default activation of grid 0=No change, 1=Grid disabled, 2=Grid enabled
horLock	String	→	Horizontal lock key
vertLock	String	→	Vertical lock key

### Description

The DR SET GLOBAL PREFERENCES command allows setting several options that will be applied to all 4D Draw areas during the work session.

- write and read: In 4D Server, templates can be stored on either the client or the server (default). The write and read parameters allow you to set the read and write locations for templates for a 4D Server database.

write is the location to which to save templates. 4D Draw uses this preference when the user chooses Save as a Template from the 4D Draw File menu, or when you call this menu item (1006) using the DR DO COMMAND command.

read is the location from which to read templates. 4D Draw uses this parameter when loading templates for 4D Draw areas on forms.

**Note:** These parameters have no effect on the single-user version of 4D Draw.

- selection: This parameter modifies the way in which objects can be selected in 4D Draw areas.
  - If you pass 1, only objects that are entirely included within the selection rectangle will be selected (default behavior).
  - If you pass 2, all the 4D Draw objects touched by the selection rectangle will be selected. This behavior is identical to that of the Form editor in 4th Dimension.

If you do not want to modify this setting, pass 0.

- **grid:** This parameter is used to set the initial state of the grid for all the 4D Draw documents that are opened or created.

- If you pass 1, the grid is disabled (default behavior).

- If you pass 2, the grid is enabled.

If you do not want to modify this setting, pass 0.

- **horLock and vertLock:** These parameters let you designate the horizontal and vertical lock keys that can be used when creating objects. Pressing one of these keys while you are drawing an object locks the object size associated with the key.

By default, the **h** and **v** keys are used respectively for horizontal and vertical locking.

**See Also**

DR GET GLOBAL PREFERENCES.

---

DR SET PREFERENCES (area; order; pict; lockAlerts; autoScroll; variable; confirm{; saveMethod})

Parameter	Type		Description
area	Longint	→	4D Draw area
order	Integer	→	Print direction
pict	Integer	→	0=Picture 1=Objects -1=No change
lockAlerts	Integer	→	0=Off 1=On -1=No change
autoScroll	Integer	→	0=Off 1=On -1=No change
variable	Integer	→	0=Fixed 1=Variable -1=No change
confirm	Integer	→	0=Without confirmation 1=With confirmation -1=No change
saveMethod	Integer	→	0=Picture and data 1=Picture only 2=Data only -1=No change

### Description

The command DR SET PREFERENCES sets several functional parameters for area, which correspond to the features that can be set in the Preferences dialog box.

order sets the order in which pages in a multipage document print when area is printed.

- If order equals 0, pages print from top to bottom, then left to right.
- If order equals 1, pages print from left to right, then top to bottom.

The default is to print pages left to right.

pict controls how PICT objects and documents are interpreted in area when they are opened, imported, or pasted.

- If pict equals 0, the PICT document or pasted PICT object is interpreted as a single object.
- If pict equals 1, 4D Draw tries to break the PICT object into its component objects.

The default option is to paste a PICT drawing as a single picture.

lockAlerts determines whether area displays an alert when a user tries to alter a locked attribute.

- If lockAlerts equals 0, no alert is presented.
- If lockAlerts equals 1, the alert is displayed.

The default is to display the alert.

autoScroll determines whether the document in area scrolls automatically when the user drags past the area boundary.

- If autoScroll equals 0, auto scrolling is turned off.
- If autoScroll equals 1, auto scrolling is turned on.

The default is to use auto-scrolling.

variable determines whether area prints using the print variable frame option.

- If variable equals 0, area prints the same size as it was defined and only those objects in the upper left corner of the document are included.
- If variable equals 1, area expands vertically to print all objects on the left side of the document.

The default is to print with a variable frame.

The confirm parameter determines if a confirmation dialog box appears when you accept a record that contains a non-autosaved 4D Draw area. By default, a confirmation dialog box appears, asking whether or not you would like to save the document in the 4D Draw area.

- If confirm equals 0, the confirmation dialog box does not appear.
- If confirm equals 1, the confirmation dialog box appears.

The optional saveMethod parameter controls how the document in the 4D Draw area is saved.

- If saveMethod equals 0, the default value, both the picture and the internal data used to rebuild the image are saved.
- If saveMethod equals 1, only the picture (PICT) is saved. Objects can no longer be manipulated individually.
- If saveMethod equals 2, only the data concerning the objects in the 4D Draw area is saved. The image is later rebuilt using the information in the saved data. This save option is the quickest and uses the least amount of memory. If there is not enough memory for the save method chosen, a dialog box that allows you to choose another method appears.

### Example

The following example turns off the display options, locked alerts, and auto scrolling for area in the On load event of a form method, without affecting the printing:

```
    If (Form event=On load)
        DR DISPLAY OPTIONS (Area;-1;0)
⇒    DR SET PREFERENCES (Area;-1;-1;0;0;-1;-1;-1)
    End if
```

### See Also

DR GET PREFERENCES.

# 4

---

## DR Areas





---

DR AREA TO AREA (source; destination; scope)

Parameter	Type		Description
source	Longint	→	Source 4D Draw area
destination	Longint	→	Destination 4D Draw area
scope	Longint	→	1=Settings 2=Objects 3=Both

### Description

The command DR AREA TO AREA copies the contents of the 4D Draw area source into the 4D Draw area destination based on the scope parameter.

- If scope equals 1, the document settings such as drawing size, ruler settings, and display options are transferred.
- If scope equals 2, all objects in source are transferred to destination.
- If scope equals 3, both objects and document settings are transferred to destination.

When document settings are transferred, they replace the document settings in destination. When objects are transferred they are appended to the objects in destination. DR AREA TO AREA is especially useful for manipulating offscreen areas.

### Example

The following example copies the contents of the 4D Draw area Diagram into a new offscreen area.

```
vOffscreen:=DR New offscreen area
⇒ DR AREA TO AREA(Diagram;vOffscreen;3)
```

### See Also

DR DELETE OFFSCREEN AREA, DR New offscreen area.

---

DR AREA TO FIELD (area; scope; table; field{; saveMethod})

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2=Document and setting saved -1=All objects saved 0=Selected object saved >0=ID
table	Integer	→	Table number
field	Integer	→	Field number
saveMethod	Integer	→	0=Picture and data 1=Picture only 2=Data only -1=No change

### Description

The command DR AREA TO FIELD copies the contents of area into the Picture field specified by table and field. If field is not of type picture, DR Error returns error code 31.

scope controls what is copied.

- If scope equals -2, the entire document is saved in the Picture field specified by table and field. This includes document settings such as drawing size, ruler settings, and display options.
- If scope equals -1, all objects in area are saved in the Picture field, but without document settings.
- If scope equals 0, only selected objects are saved in the Picture field.
- If scope is greater than 0, it must equal an object's ID, and only that object is saved. If an object with that ID does not exist, DR Error returns error code 2.

DR AREA TO FIELD is useful when you want to store objects in a field of a related table or store only specific objects. DR AREA TO FIELD simply assigns the objects to field. The record in table must still be saved.

The optional saveMethod parameter determines how the document in the 4D Draw area is saved.

- If saveMethod equals 0, the default value, both the picture and the internal data used to rebuild the image are saved.
- If saveMethod equals 1, only the picture (PICT) is saved. Objects can no longer be manipulated individually.
- If saveMethod equals 2, only the data concerning the objects in the 4D Draw area is saved. The image is later rebuilt using the information in the saved data. This save option is quick and uses the least amount of memory. If there is not enough memory, a dialog box that allows you to choose another method appears.

## Example

The following example creates a related record for each object in Area.

```
For($i;1;DR Count(Area;-1))
  `Loop for each object
  CREATE RECORD([Objects])
  `Create a record to store the object
  [Objects]Key:=[Drawings]Name
  `Assign the relating value
  $Temp:=DR Get ID (Area;-1;$i)
  `Get the object's ID
⇒ DR AREA TO FIELD(Area;$Temp;3;2;1)
  `Copy the object into the record
  SAVE RECORD([Objects])
  `Save the record
End for
```

## See Also

DR FIELD TO AREA.

DR Area to picture (area; scope) → Picture

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2=Document -1=All 0=Selected >0=ID
Function result	Picture	←	4th Dimension picture of objects in area

### Description

The command DR Area to picture returns a 4th Dimension picture of objects in area.

The objects included in the picture are controlled by the scope parameter.

- If scope equals -2, the entire document is copied. This includes document settings such as drawing size, ruler settings, and display options.
- If scope equals -1, all objects in area are copied but without document settings.
- If scope equals 0, only selected objects are copied.
- If scope is greater than 0, it must equal a specific object's ID and only that object is copied. If the object does not exist, DR Error returns error code 2.

### Example

The following example creates a picture array of the selected objects in Area. Each object becomes one element in the array.

```

$x := DR Count (Area;0)
  `Count the number of selected objects
ARRAY PICTURE (aPict;$x)
  `Declare picture array to hold objects
For ($i;1;$x)
  `Loop for each selected object
  $ID := DR Get ID (Area;0;$i)
  `Get the selected object's ID
⇒ aPict{$i} := DR Area to picture (Area;$ID)
  `Put object into array element
End for

```

### See Also

DR Get ID, DR PICTURE TO CLIPBOARD, DR PLACE PICTURE.

---

DR DELETE OFFSCREEN AREA (area)

Parameter	Type		Description
area	Longint	→	4D Draw area

### Description

The command DR DELETE OFFSCREEN AREA disposes of a 4D Draw offscreen area that was created with DR New offscreen area and frees the memory used.

area must be an offscreen area rather than an area on a form or in a window. You should always call DR DELETE OFFSCREEN AREA when you are finished with an offscreen area.

### Example

The following example illustrates how to pair every call to DR New offscreen area with a corresponding call to DR DELETE OFFSCREEN AREA.

```

    $NewArea := DR New offscreen area
      `Create a new offscreen area
      `Do some processing here
⇒ DR DELETE OFFSCREEN AREA ($NewArea)
      `Remove the offscreen area
  
```

### See Also

DR AREA TO AREA, DR New offscreen area.

DR FIELD TO AREA (area; table; field)

Parameter	Type		Description
area	Longint	→	4D Draw area
table	Integer	→	Table number
field	Integer	→	Field number

### Description

The command DR FIELD TO AREA places into area the drawing contained in the Picture field specified by table and field.

field must be of type picture. field may contain either a previously saved 4D Draw drawing or a picture. The contents of field replaces the contents of area. If field is empty, the command is ignored, and DR Error returns error code 39.

### Example

The following example opens the 4D Draw drawing contained in the fifth field of the second table.

```
⇒      If (Before)
           DR FIELD TO AREA (Area;2;5)
           End if
```

### See Also

DR AREA TO FIELD.

DR NEW DRAWING (area)

Parameter	Type		Description
area	Longint	→	area

### Description

The command DR NEW DRAWING clears the contents of the drawing in area. This command is equivalent to choosing **New** from the **File** menu, except that no confirmation dialog box is presented. It clears all objects and all document settings, such as document size and ruler scaling.

### Example

The following example clears the drawing in area.

⇒ *DR NEW DRAWING* (area)

When you use this command, the current drawing in area is not saved. To save the current drawing, you must call the DR SAVE DOCUMENT command before calling the DR NEW DRAWING command.

### See Also

DR SAVE DOCUMENT.

DR New offscreen area → Longint

Parameter	Type	Description
This command does not require any parameters		
Function result	Longint	← Area's ID

### Description

The command DR New offscreen area creates a 4D Draw offscreen area and returns the area's ID. The value returned by DR New offscreen area can be used in any 4D Draw command that requires a 4D Draw area.

### Example

This example searches for a record, creates an offscreen area, copies a drawing from the record into the area, and then prints the area.

```

    QUERY ([Table3];[Table3]Field1 = "Level1")
      `Search for the record
⇒  $Offscreen := DR New offscreen area
      `Create a new offscreen area
    DR FIELD TO AREA ($Offscreen;3;2)
      `Copy the drawing stored in a field
    DR PRINT ($Offscreen;0)
      `Print the area
    DR DELETE OFFSCREEN AREA ($Offscreen)
      `Get rid of the offscreen area

```

### See Also

DR AREA TO AREA, DR DELETE OFFSCREEN AREA.



---

DR OPEN DOCUMENT (area; document; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
document	String	→	Name of document, 255 chars maximum
mode	Integer	→	0=Replace 1=Combine

### Description

The command DR OPEN DOCUMENT opens document and places its contents into area. The document must be of type 4D Draw (\*.4dw on Windows), PICT (\*.pct on Windows), EPSF (\*.eps on Windows) or MacPaint PNTG (\*.pnt on Windows). If document is an empty string, DR OPEN DOCUMENT displays the standard Open File dialog box, enabling the user to choose the document. If document does not exist, the contents of area remain unchanged and DR Error returns a system error code. If the document is already open, DR Error returns error code 43.

4D Draw expects to find document in the folder that contains the database structure. If you want to open a document outside of the database folder, specify a complete pathname. If document is already open, DR Error returns a Macintosh system error.

- **Windows:** Enter the letter of the disk plus a colon (":") and then a "\" between each folder. The file name must have an extension to determine its type, for example:

"D:\Folder1\Folder2\File.PCT".

- **MacOS:** Enter the name of the disk and a ":" between each folder, for example: "Hard Disk:Folder:Document".

The optional mode parameter controls how the document is opened. mode is used only when document is not an empty string and is recognized only for documents of type PICT, PNTG, and EPSF.

If mode equals 0 or is not specified, document replaces the contents of area.

If mode equals 1, document is combined with the current contents of area.

## Example

This example opens a different drawing based on the value of the Client Type field.

```
Case of
  : ([Client]Client Type="Distrib.4DW")
    `If the type is Distrib.4DW
⇒   DR OPEN DOCUMENT (Area;"Distrib.4DW")
    `Open the Distrib.4DW drawing
  : ([Client]Client Type="Construc.4DW")
    `If the type is Construc.4DW
⇒   DR OPEN DOCUMENT (Area;"Construc.4DW")
    `Open Construc.4DW drawing
  : ([Client]Client Type="Client.4DW")
    `If the type is Client.4DW
⇒   DR OPEN DOCUMENT (Area;"Client.4DW")
    `Open the Client.4DW drawing
End case
```

## See Also

DR SAVE DOCUMENT.

---

DR PICTURE TO AREA (area; picture)

Parameter	Type		Description
area	Longint	→	4D Draw area
picture	Picture	→	Picture

### Description

The command DR PICTURE TO AREA places into area the drawing contained in picture.

picture must be a valid 4th Dimension picture expression. picture can contain either a previously saved 4D Draw drawing or a picture. The contents of area are replaced with picture. If picture is empty, the command is ignored and DR Error returns error code 39.

### Example

The following example sets up the current 4D Draw document based on a template picked from an element in an array:

```
QUERY([Templates];[Templates]Type=2)
SELECTION TO ARRAY([Templates]DrawDoc;aTemplates)
```

The following is the object method for the aTemplates array:

```
CONFIRM("Replace current 4D Draw area?")
If (OK=1)
⇒  DR PICTURE TO AREA(Area;aTemplates{aTemplates})
End if
```

### See Also

DR Area to picture.

## Example

This example opens a different drawing based on the value of the Client Type field.

```
Case of
  : ([Client]Client Type="Distrib.4DW")
    `If the type is Distrib.4DW
⇒   DR OPEN DOCUMENT (Area;"Distrib.4DW")
    `Open the Distrib.4DW drawing
  : ([Client]Client Type="Construc.4DW")
    `If the type is Construc.4DW
⇒   DR OPEN DOCUMENT (Area;"Construc.4DW")
    `Open Construc.4DW drawing
  : ([Client]Client Type="Client.4DW")
    `If the type is Client.4DW
⇒   DR OPEN DOCUMENT (Area;"Client.4DW")
    `Open the Client.4DW drawing
End case
```

## See Also

DR SAVE DOCUMENT.

---

DR SAVE DOCUMENT (area; document; type; scope)

Parameter	Type		Description
area	Longint	→	4D Draw area
document	Text	→	Name of document
type	String	→	Type of document
scope	Longint	→	0=All objects 1=Selected objects

### Description

The command DR SAVE DOCUMENT saves the contents of area into document.

If document is an empty string, DR SAVE DOCUMENT displays the standard Save File dialog box, enabling the user to specify the document name, type, and scope. If document is not an empty string, DR SAVE DOCUMENT saves document with the type Type. If document does not exist, DR SAVE DOCUMENT creates it. If document exists, DR SAVE DOCUMENT overwrites it.

If type is an empty string, a standard 4D Draw document is created.

Three saving formats can be passed to type as a string; which can have three or four characters on either platform. The four-character format maintains compatibility with the old MacOS version.

The formats are:

- 4DW or 4DRW
- PCT or PICT
- PNT or PNTG

The optional scope parameter controls what is saved into document. Use scope only when document is not an empty string and when working with documents of type PICT and PNTG.

- If scope equals 1, only selected objects are saved. If no objects are selected, the document is not saved, and DR Error returns error 37.
- If scope equals 0, or if it is missing, all objects in area are saved.

By default, document is saved in the folder that contains the database structure. To save a document outside of the database folder, specify a complete pathname.

• **Windows:** Enter the letter of the disk plus a colon (":") and then a "\" between each folder. The file name must have an extension to determine its type, for example:

"D:\Folder1\Folder2\File.PCT".

• **MacOS:** Enter the name of the disk and a ":" between each folder, for example: "Hard Disk:Folder:Document".

If the document is already open, DR Error returns error code -49.



5

---

# DR Binding





The Binding theme commands and functions enable you to create and work with binds between objects in a 4D Draw area and fields in a database.

Object binding is a unique feature in 4D Draw that lets you establish a connection between objects in drawings and records in a database. Binding provides a way to associate the attributes of objects with field values. You can get the same results by using a series of 4D Draw and 4th Dimension commands to test and modify object attributes and field values. A bind, however, requires less code, is simpler to implement, and is more easily modified.

You can create any number of binds, but only one bind can be active for a given 4D Draw area at a time. To use a bind, you must first specify which attributes are to be bound to which fields. For more information, see the DR ADD TO BIND command.

Once a bind is activated, the fields take on the values of the attributes of the selected objects. If no objects are selected, the bound fields show the default values for the area. If only one object is selected, the bound fields show the values for that object. If more than one object is selected, the bound fields show the values that the objects have in common. If the objects are not the same for a given attribute, or if the attribute is inappropriate for the selected object, the bound field displays an error code. The error code is -32000, or another code, depending on the field type.

When the user makes an entry in a bound field, the objects' attributes are updated. If no objects are selected, the default values are set. If multiple objects are selected, they are all updated.

Some attributes cannot be modified. If the user attempts to modify a field bound to a non-modifiable attribute, the field's value is reset to its former value. Fields that are bound to non-modifiable attributes should be non-enterable. See Appendix A, Attribute Codes for a complete list of attribute codes and corresponding field types.

Much of the information used in a bind is numeric. For example, a field bound to the fill pattern returns the number of the pattern in the palette. To determine the meaning of the values in the bound fields, see the commands that affect that attribute. For example, to determine how to interpret or set a fill pattern, see the DR GET FILL ATTRIBUTES or DR SET FILL ATTRIBUTES command.

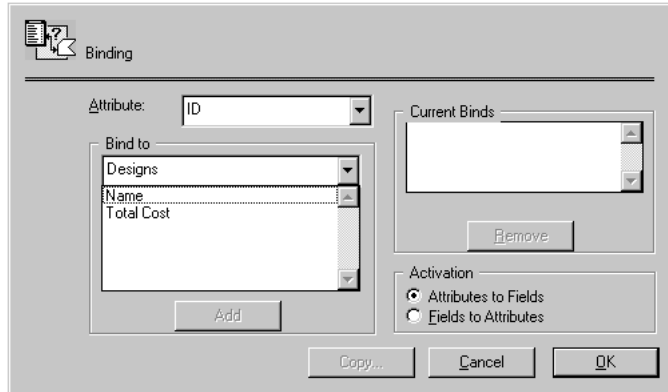
Coordinate and size attributes are specified in scale units according to the origin. This matches what the user sees in the Coordinates panel. Use the DR Scale to base function to convert from scale units to base units.

Binding always affects the values in the current record. If there is no current record, the bind has nothing to affect.

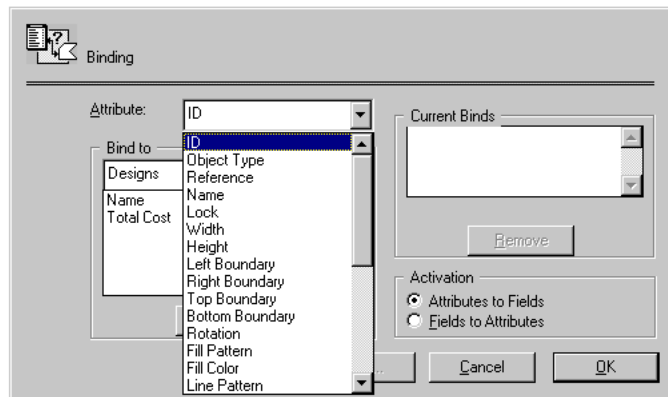
To create a binding method in the User environment:

1. Select the object(s) for which you want to activate a bind.
2. Choose **Binding** from the Database menu.

The Binding dialog box appears.

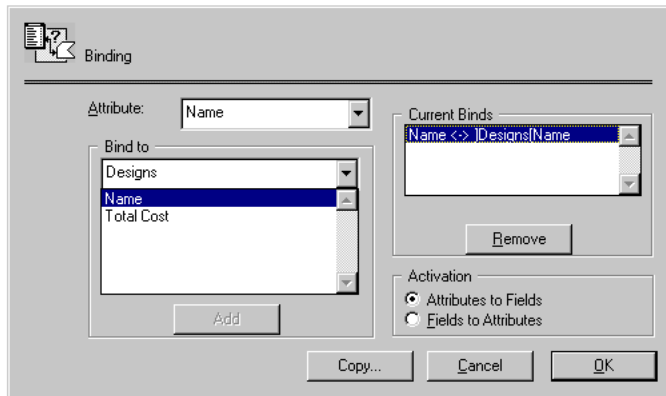


3. Choose an attribute from the Attribute pop-up menu.



4. Choose a table from the pop-up menu of tables.  
The fields for the selected table are displayed.

5. Choose a field to which to bind the 4D Draw attribute.



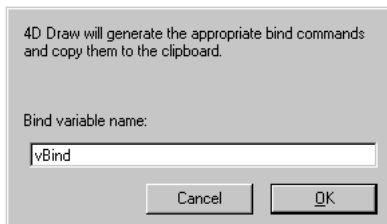
6. Choose a bind direction from the Activation area:

- **Attributes to Fields:** When you modify the objects in the 4D Draw area, the value of the field changes.
- **Fields to Attributes:** When you enter a value in the field, the objects in the 4D Draw area are modified.

7. Click Add to add the bind to the Current Binds list.

The bind will be activated when you click the OK button. If you want to use the bind within your database, you must copy the bind code into a method in the Design environment.

8. Click the Copy button to copy the binding method to the Clipboard.  
When you click the Copy button, the following dialog box appears:



9. Enter the variable name you wish to use for the bind and click OK.

This variable name is used in the commands 4D Draw creates. The commands are copied to the Clipboard, from which you can paste them directly into a method window.

The bind information copied to the Clipboard defines the bind you created in the dialog box. For instance, the Clipboard might contain the following lines:

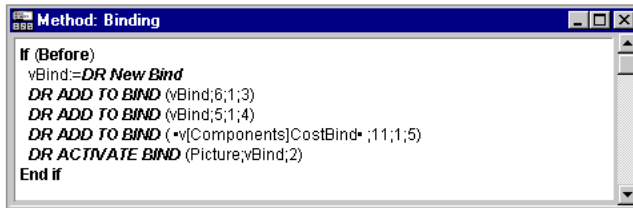
```
vBind := DR New Bind  
DR ADD TO BIND (vBind;6;1;3)  
DR ADD TO BIND (vBind;5;1;4)  
DR ADD TO BIND (vBind;11;1;5)
```

To make the method work, you must add an additional line of code to activate the bind. The statement that activates the bind specifies whether the bind should go from the drawing to the fields or from the fields to the drawing.

In the following method, the statement:

```
DR ACTIVATE BIND (Picture;vBind;2)
```

is added to activate the bind. The last parameter to the command indicates that the bind should go from the fields to the drawing.



```
Method: Binding  
If (Before)  
vBind:=DR New Bind  
DR ADD TO BIND (vBind;6;1;3)  
DR ADD TO BIND (vBind;5;1;4)  
DR ADD TO BIND (*v[Components]CostBind* ;11;1;5)  
DR ACTIVATE BIND (Picture;vBind;2)  
End if
```

---

DR ACTIVATE BIND (area; bindID; direction)

Parameter	Type		Description
area	Longint	→	4D Draw area
bindID	Longint	→	Bind ID
direction	Integer	→	1=Drawing ->Fields 2=Fields ->Drawing

### Description

The command DR ACTIVATE BIND associates the bind described by bindID with area.

bindID is the value returned by the DR New bind function.

A bind can work in either of two directions: drawing to fields, or fields to drawing. If the bind is activated in the drawing to fields direction, any changes to the selected objects in area result in changes to the bound fields. If a bind is activated in the fields to drawing direction, any changes to the bound fields result in corresponding changes to the attributes of selected objects in area.

The direction parameter controls what happens when the bind is activated. If direction equals 1, the bound fields are updated based on the selected objects. If direction equals 2, the objects are updated based on the bound fields. After the bind is activated, it will work in both directions.

You cannot delete a bind that is currently active, and you cannot have more than one bind in effect at one time for a given area. A bind stays active until deactivated or until the area is closed. To associate an area with a different bind, you must first deactivate the active bind using the DR DEACTIVATE BIND command.

### Example

The following example shows how to build and activate a bind.

```

vBind := DR New bind
DR ADD TO BIND (vBind;0;1;1) `Bind object ID to the first field
DR ADD TO BIND (vBind;5;1;2) `Bind object width to second field
DR ADD TO BIND (vBind;6;1;3) `Bind object height to the third field
DR ADD TO BIND (vBind;11;1;4) `Bind object rotation to fourth field
⇒ DR ACTIVATE BIND (Area;vBind;1) `Activate the bind

```

### See Also

DR DEACTIVATE BIND.

---

DR ADD TO BIND (bindID; attribute; table; field)

Parameter	Type		Description
bindID	Longint	→	Bind ID
attribute	Longint	→	Attribute code
table	Integer	→	Table number
field	Integer	→	Field number

### Description

The command DR ADD TO BIND associates the object attribute described by attribute with the field described by table and field for the bind specified by bindID. bindID is the value returned by the DR New bind function.

Use this command once for each attribute that you want to associate with a field. There are 27 different object attributes that can be bound. No attribute or field can be used twice in the same bind

To change the attribute bound to a given field, first remove the attribute/field with the DR REMOVE FROM BIND command and re-bind it with a new call to DR ADD TO BIND. A bind must contain at least one attribute/field pair to be activated. Once a bind is active (associated with an area), adding new attributes will not affect the area unless the bind is deactivated and then reactivated.

Each attribute needs to be bound to a field of a specific type. If the attribute is not bound to the correct field type, it will not function properly. In some cases if you bind an attribute to a field that has a more constrained type (i.e., Long Integer instead of Real) the values are truncated. See Appendix A, Attribute Codes for a complete list of attribute codes and corresponding field types.

**Note:** Some attributes, such as the object ID, cannot be modified.

If the field bound to an attribute is given an inappropriate value (such as a negative width), the field is reset to its previous value.

### Example

See the example for the DR ACTIVATE BIND command.

### See Also

DR REMOVE FROM BIND.

DR DEACTIVATE BIND (area)

Parameter	Type		Description
area	Longint	→	4D Draw area

### Description

The command DR DEACTIVATE BIND deactivates the bind associated with area. A bind must be deactivated and then reactivated before changes to the bind take effect. Use this command to deactivate the bind in order to change or delete it.

### Example

The following example demonstrates how to modify a bind for an area.

```
⇒ DR DEACTIVATE BIND (Area)
   DR ADD TO BIND (vBind;12;1;5)
   DR ACTIVATE BIND (Area;vBind;1)
```

### See Also

DR ACTIVATE BIND.

DR DELETE BIND (bindID)

Parameter	Type		Description
bindID	Longint	→	Bind ID

### Description

The command **DR DELETE BIND** disposes of a bind and frees the memory used. Binds use little memory, but a complex database may hold many binds. Use this command to free the memory used by a bind. A bind cannot be deleted while it is active.

### Example

The following example shows how to delete a bind once it is no longer needed.

```
vBind := DR New bind
    `Create a new bind
    `Do some processing here
⇒ DR DELETE BIND (vBind)
    `Delete the bind and free up memory
```

### See Also

DR New bind.



---

DR New bind → Longint

Parameter	Type		Description
This command does not require any parameters			
Function result	Longint	←	New bind ID

### Description

The command DR New bind creates a new bind in memory and returns a value that can be used to access the new bind. The value returned is referred to as a Bind ID and is used in all of the other binding commands. Bind IDs are long integers. A bind is not specific to a 4D Draw area.

To build a bind, use the DR ADD TO BIND command for each attribute you want to associate with a field.

### Example

See the example for the DR ACTIVATE BIND command.

### See Also

DR DELETE BIND.

DR REMOVE FROM BIND (bindID; attribute)

Parameter	Type		Description
bindID	Longint	→	Bind ID
attribute	Longint	→	Object attribute

### Description

The command DR REMOVE FROM BIND removes the object attribute described by attribute from the bind described by bindID. Use this command to remove an attribute so that it can be bound to a different field.

Once a bind is active (associated with an area), removing attributes does not affect the area unless the bind is deactivated and then reactivated.

See Appendix A, Attribute Codes for a complete list of attribute codes and corresponding field types

### See Also

DR ADD TO BIND.

# 6

---

## DR Get Attributes



---

DR GET ARC SPECS (area; scope; start; length; radiusH; radiusV; centerH; centerV)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Object ID
start	Integer	←	Starting point angle in degrees
length	Integer	←	Arc length in degree
radiusH	Number	←	Horizontal radius
radiusV	Number	←	Vertical radius
centerH	Number	←	Horizontal location of arc center
centerV	Number	←	Horizontal location of arc center

### Description

The command DR GET ARC SPECS returns into the start, length, radiusH, radiusV, centerH, and centerV variables various specifications about the arc in area described by scope.

- If scope equals -1, the command returns specifications for the first arc in the document.
- If scope equals 0, the command returns the specifications for the first selected arc.
- If scope is greater than 0, it must be equal to a specific arc's object ID and the command returns that arc's specifications. If the object does not exist, DR GET ARC SPECS returns -32000 for each attribute, and DR Error returns error number 2. If the object described by scope is not an arc, the command returns -32000 for each attribute, and DR Error returns error number 47.

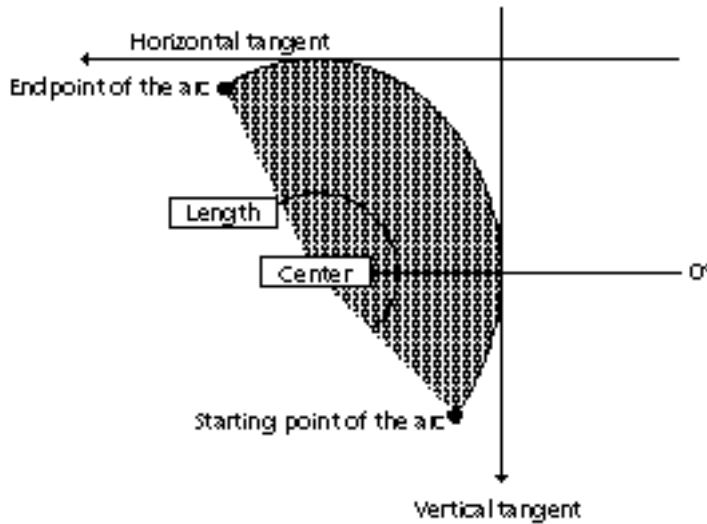
start is the beginning point of the arc in degrees. The arc is measured moving in a counter-clockwise fashion.

length is the length of the arc in degrees. The end of an arc is always start + length % (modulo) 360. Arcs have a maximum length of 359°.

radiusH and radiusV are the horizontal and vertical radii of the arc expressed in base units. These two numbers can be different, because an arc is a portion of an oval rather than a circle. If radiusH equals radiusV, the arc is a portion of a circle.

centerH and centerV are the horizontal and vertical coordinates of the center of the arc with respect to the origin; they are expressed in base units. Use the DR Base to scale function to convert from base units to scale units. See the diagram for the DR Draw arc function.

The following diagram indicates the relevant attributes of an arc:



### Example

The following example uses DR GET ARC SPECS to display an alert that displays the beginning and ending of the selected arc.

```
⇒ DR GET ARC SPECS (Area;O;vStart;vLength;vRadiusH;vRadiusV;vCenterH;vCenterV)
  If (0 = DR Error)
    ALERT("Arc starting point: "+String (vStart)+Char(13)+"Arc ending point: "
          +String(vStart+vLength))
  Else
    ALERT ("Select a single arc first")
  End if
```

### See Also

DR Draw arc, DR SET ARC SPECS.

DR Get attribute lock (area; scope; attributeNum) → Integer

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All, 0=Selected objects, >0=Object ID
attributeNum	Integer	→	Number of the attribute
Function result	Integer	←	State of the attributeNum parameter 0 = Not locked, 1 = Locked

### Description

The command DR Get attribute lock returns the state of the attributeNum parameter. The codes that can be passed in attributeNum are listed below.

- If DR Get attribute lock returns 0, attributeNum is not locked.
- If DR Get attribute lock returns 1, attributeNum is locked.

Here is the list of lock attribute codes:

Attributes	Values
<b>General</b>	
Name	3
Deletion	0
Location	7
Size	5
Shape	25
Rotation	11
Ungrouping	1
Corner Rounding	24
<b>Text</b>	
Font	19
Size	20
Style	21
Justification	22
Edition	23
<b>Line</b>	
Color	15
Pattern	14
Size	16
Endmarks	17
<b>Fill</b>	
Color	13
Pattern	12

---

DR GET BOUNDARY (area; scope; left; top; right; bottom)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Object ID
left	Number	←	Left boundary
top	Number	←	Top boundary
right	Number	←	Right boundary
bottom	Number	←	Bottom boundary

### Description

The command DR GET BOUNDARY returns into the left, top, right, and bottom variables, the boundary of the objects in area described by scope.

The boundary is the coordinates of the smallest rectangular region that contains objects and is expressed in base units according to the current origin. Use the DR Base to scale function to convert from base units to scale units.

- If scope equals -1, the command returns the boundary for all objects in the document.
- If scope equals 0, the command returns the specifications for the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID; the command returns that object's boundary.

If the object does not exist, DR GET BOUNDARY returns -32000 for each coordinate, and DR Error returns 2.

### Example

The following example is the object method for a button on a form that contains area. When the button is pressed, the document is scrolled so that the upper left corner of the selected objects is visible.

```
⇒ DR GET BOUNDARY (Area;0;$Left;$Top;$Right;$Bottom)
   DR SCROLL DOCUMENT (Area;$Left;$Top)
```

### See Also

DR Base to scale, DR GET AREA BOUNDARY, DR Get ID, DR SET ORIGIN.



DR Get corner rounding (area; scope) → Number

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2=Default -1=All 0=Selected >0=Object ID
Function result	Number	←	Corner rounding of the objects in area described by scope

### Description

The command DR Get corner rounding returns the corner rounding of the objects in area described by scope.

The corner rounding is expressed in base units. Use the DR Base to scale function to convert from base units to scale units.

- If scope equals -2, the command returns the default corner rounding for new round rectangles.
- If scope equals -1, the command returns the corner rounding for all objects in the document.
- If scope equals 0, the command returns the corner rounding for the selected objects. If the corner rounding is not the same for all of the objects described by scope, DR Get corner rounding returns -32000, and DR Error returns error number 29.
- If scope is greater than 0, it must be equal to a specific object's ID; the command returns that object's corner rounding. If the object does not exist, DR Get corner rounding returns -32000 and DR Error returns 2.

### Example

The following example gets the corner rounding of the selected objects and then increments it by 0.1 base units.

```

$Temp := DR Get corner rounding (Area;0)
If ($Temp # -32000)
⇒   DR SET CORNER ROUNDING (Area;0;$Temp + 0.1)
End if

```

### See Also

DR Get ID, DR SET CORNER ROUNDING.

---

DR GET ENDMARKS (area; scope; type; point)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2=Default -1=All 0=Selected >0=Object ID
type	Integer	←	1=Arrow 2=Bar
point	Integer	←	0=None 1=Start 2=End 3=Both

### Description

The command DR GET ENDMARKS returns into the integer variables type and point, the values defining the endmarks of the lines in area described by scope.

- If scope equals -2, the command returns the default endmarks for new lines.
- If scope equals -1, the command returns the endmarks for all lines in the document.
- If scope equals 0, the command returns the endmarks for the selected lines.
- If scope is greater than 0, it must be equal to a specific line's ID; the command returns that line's endmarks. If the object does not exist, DR GET ENDMARKS returns -32000 for each attribute, and DR Error returns error number 2.

type indicates the kind of endmark on the specified line. Every line has a type of endmark, even though the endmarks may not be displayed.

- If type equals 1, the lines have arrow endmarks.
- If type equals 2, the lines have bar endmarks.

point indicates which ends of the line have endmarks.

- If point equals 0, the lines have no endmarks showing.
- If point equals 1, the lines have endmarks at the beginning.
- If point equals 2, the lines have endmarks at the end.
- If point equals 3, the lines have endmarks on both ends.

If type and point are not the same for all of the lines described by scope, DR GET ENDMARKS returns -32000 for that parameter and DR Error returns error number 29.

## Example

The following example gets the ID and the type of the selected object, and if the object is a line, opens an alert that displays which ends of the line have endmarks.

```
$Temp := DR Get ID (Area;0;1)  `Get the ID of the 1st selected object
$Type := DR Get object type (Area;$Temp)  `Get the type of $Temp
If ($Type = 9)  `If the object is a line
⇒   DR GET ENDMARKS (Area;$Temp;vKind;vPoint)  `Get its endmarks
      Case of
        :(vPoint = 1)  `Endmark at the start
          ALERT ("This line has an endmark at the start.")
        :(vPoint = 2)  `Endmark at the end
          ALERT ("This line has an endmark at the end.")
        :(vPoint = 3)  `Endmarks at the start and end
          ALERT ("This line has an endmark at both the start and the end.")
        Else  `No endmarks
          ALERT ("This line has no endmarks.")
      End case
    End if
```

## See Also

DR SET ENDMARKS.

---

DR GET FILL ATTRIBUTES (area; scope; pattern; color)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2=Default -1=All 0=Selected >0=ID
pattern	Integer	←	Pattern index
color	Longint	←	Color value

### Description

The command DR GET FILL ATTRIBUTES returns into the pattern and color variables the fill attributes for the objects in area described by scope.

- If scope equals -2, the command returns the default fill attributes.
- If scope equals -1, the command returns the fill attributes for all objects in the document.
- If scope equals 0, the command returns the fill attributes for the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID, the command returns that object's attributes. If the object does not exist, DR GET FILL ATTRIBUTES returns -32000 for each attribute, and DR Error returns 2.

The codes for the scope parameter are given in the paragraph "Specifying the Scope of a Command" of the Manipulating Objects section.

pattern is the number of the pattern in the palette. Patterns have a value in the range 1 to 36 and are numbered left to right, top to bottom.

color is a long integer that represents the color of the object's interior. This number can be used in the commands DR SET FILL ATTRIBUTES and DR SET LINE ATTRIBUTES to set other objects to the same color.

If any of the attributes are not the same for all of the objects described by scope, DR GET FILL ATTRIBUTES returns -32000 for that attribute.

## Example

The following example requests the user for an object's ID, checks to make sure the ID is valid, gets the object's fill pattern and color, and calls a project method that changes the pattern and color to emphasize the object.

```
$Temp := Num (Request ("Which object?";"1"))
  `Get an objects ID number
If (OK=1) & ($Temp > 0)
  `If they give a valid number
⇒  DR GET FILL ATTRIBUTES (Area;$Temp;vPattern;vColor)
  `Get the fill attributes
  If (vPattern # -32000)
    `If the object exists
    HIGHLIGHT (Area;$Temp;vPattern;vColor)
    `Pass info to a project method
  Else
    `Otherwise...
    ALERT("There is no object with that ID!")
    `Let the user know
  End if
End if
```

## See Also

DR SET FILL ATTRIBUTES.

## DR Get handle state

## DR Get Attributes

version 6.0

---

DR Get handle state (area; scope) → Integer

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All objects 0=Selected >0=ID

### Description

The command DR Get handle state returns **0** or **1**, depending on whether or not the handles for the objects in scope are visible.

---

DR GET HIGHLIGHT (area; first; last)

Parameter	Type		Description
area	Longint	→	4D Draw area
first	Longint	←	Position of first character minus 1
last	Longint	←	Position of last character

### Description

The command DR GET HIGHLIGHT returns into the first and last variables the character positions of the highlighted text in area.

first is one less than the first character position highlighted, and last is the last character highlighted. If first equals last, no characters are highlighted, and the insertion point is between first and first +1.

As only one object at a time can have highlighted text, the scope parameter is not needed. If there is no highlighted text in area, DR GET HIGHLIGHT returns -32000 for first and last, and DR Error returns 56.

### Example

The following example returns the position of the highlighted text and if no text is selected it alerts the user.

```
⇒  DR GET HIGHLIGHT (Area;$First;$Last)
    If (DR Error=56)
      ALERT ("There is no text highlighted.")
    End if
```

### See Also

DR SET HIGHLIGHT.

---

DR Get ID (area; scope; index) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Group ID
index	Longint	←	Index number into list
Function result	Longint	←	Object ID

### Description

The command DR Get ID returns the unique ID for the object in area described by scope and index. This number is used by many other 4D Draw commands and is referred to as an object's ID.

To get an object's ID, you specify the set of objects to which to refer; then specify the order of the object within the set. Objects are ordered from back to front. The object that is farthest in back has an index of 1.

- If scope equals -1, then index refers to the order of the object within the entire document.
- If scope equals 0, then index refers to the order of the object within the currently selected objects.
- If scope is greater than 0, it must be the ID for a group, and index refers to the order of objects within the group. This last syntax lets you manipulate objects in a group without ungrouping.

The codes for the scope parameter are given in the paragraph “Specifying the Scope of a Command” of the Manipulating Objects section.

### Examples

(1) The following example shows how to extract the ID for an object.

```
⇒ vID := DR Get ID (Area;0;1)
   `Get the ID of the first selected object
```



(2) The following code segments show how to loop through different sets of objects and get their ID's.

```
    `Looping through all objects
    `Counts the number of objects in the entire document
    $Count := DR Count (Area;-1)
    For($i;1;$Count)
    ⇒    `Loop for all objects in the document
        vObjectID := DR Get ID (Area;-1;$i)
        `Get the object's ID
        ...
        `Do something for each object here
    End for
    `Looping through selected objects
    $Count := DR Count (Area;0)
    `Count the selected objects
    For($i;1;$Count)
    ⇒    `Loop for all selected objects
        vObjectID := DR Get ID (Area;0;$i)
        `Get the object's ID
        ...
        `Do something for each object here
    End for
    `Looping though objects in a group
    $Count := DR Count (Area;vGroupID)
    `Count the objects in the group
    For($i;1;$Count)
    ⇒    `Loop for all objects objects in group
        vObjectID := DR Get ID (Area;vGroupID;$i)
        `Get the object's ID
        ...
        `Do something for each object here
    End for
```

**See Also**

DR Count.

---

DR GET LINE ATTRIBUTES (area; scope; pattern; color; width)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2=Default, -1=All, 0=Selected, >0=Group ID
pattern	Integer	←	Pattern index
color	Longint	←	Color value
width	Number	←	Line width in points

### Description

The command DR GET LINE ATTRIBUTES returns into the pattern, color, and width variables the line attributes for the objects in area described by scope.

- If Scope equals -2, the command returns the default line attributes.
- If Scope equals -1, the command returns the line attributes for all objects in the document.
- If Scope equals 0, the command returns the line attributes for the selected objects.
- If Scope is greater than 0, it must be equal to a specific object's ID; the command returns that object's attributes. If the object does not exist, DR GET LINE ATTRIBUTES returns -32000 for each attribute, and DR Error returns error number 2.

pattern is the number of the pattern in the palette. Patterns have a value in the range 1 to 36 and are numbered left to right, top to bottom.

color is a long integer that represents the color of the lines. This number can be used in the commands DR SET FILL ATTRIBUTES and DR SET LINE ATTRIBUTES to set other objects to the same color.

width is a real number that describes the width (thickness) of the line in points (1/72 of an inch).

If any of the attributes are not the same for all of the objects described by scope, DR GET LINE ATTRIBUTES returns a -32000 for that attribute.

### Example

The following example checks the default pattern, color and width of lines and if the line attributes are not solid, black, and 0.25 points wide, it resets them.

```
⇒ DR GET LINE ATTRIBUTES (Area;-2;vPattern;vColor;vWidth)
   If (vPattern # 3) | (vColor # 0) | (vWidth # 0.25)
     DR SET LINE ATTRIBUTES (Area;-2;3;0;0.25)
   End if
```

### See Also

DR SET LINE ATTRIBUTES.

---

DR GET LINE SPECS (area; scope; startH; startV; endH; endV)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Object ID
startH	Number	←	Horizontal position of start
startV	Number	←	Vertical position of start
endH	Number	←	Horizontal position of end
endV	Number	←	Vertical position of end

### Description

The command DR GET LINE SPECS returns into the startH, startV, endH, and endV variables the endpoints of the line in area described by scope.

- If scope equals -1, the command returns endpoints for the first object in the document.
- If scope equals 0, the command returns the endpoints for the first selected object.
- If scope is greater than 0, it must be equal to a specific line's ID; the command returns that line's endpoints. If the object does not exist, DR GET LINE SPECS returns -32000 for each attribute, and DR Error returns error number 2. If the object described by scope is not a line, DR GET LINE SPECS returns -32000 for each attribute and DR Error returns error number 47.

startH and startV are the horizontal and vertical positions of the starting point of the line. endH and endV are the horizontal and vertical positions of the ending point of the line. When the user creates a line, the starting point is the point at which the user begins drawing the line and the end is where the mouse is released. As a consequence, the beginning of a line is not necessarily above or to the left of the end of the line. It is important to identify the beginning and ending of a line when adding endmarks.

All coordinates are expressed in base units according to the current position of the origin. Use the DR Base to scale function to convert from base units to scale units.

## Example

The following example draws an oval around the selected line and brings the line to the front of the drawing:

```
⇒ DR GET LINE SPECS (area;0;startH;StartV;EndH;endV)
   If (0=DR Error)
     Oval:=DR Draw oval (area;startH-0.25;startV-0.25;endH+0.25; endV+0.25;0)
     DR DO COMMAND (area;5001) ` Bring the line to the front
   Else
     ALERT("Please select a single line first!")
   End if
```

## See Also

DR SET ENDMARKS, DR SET LINE SPECS.

---

DR Get name (area; scope) → String

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2=Default -1=All 0=Selected >0=Group ID
Function result	String	←	Name of the objects in area

### Description

The command DR Get name returns the name of the objects in area described by scope. A name is a string associated with an object and is not necessarily unique. Names have a maximum length of 31 characters. Users can set names using the Object Attribute dialog box or using the DR SET NAME command in a method.

- If scope equals -2, the function returns the default name. The default name is usually an empty string and can only be set procedurally.
- If scope equals -1, the function returns the name for all objects in the document. If the names of the objects are not equal, DR Get name returns “\*\*\*\*\*” and DR Error returns error number 29.
- If scope equals 0, the function returns the name for the selected objects. If the names of the objects are not equal, DR Get name returns “\*\*\*\*\*” and DR Error returns error number 29.
- If scope is greater than 0, it must be equal to a specific object’s ID; the function returns that object’s name. If the object does not exist, DR Get name returns an empty string and DR Error returns error number 2.

The codes for the scope parameter are given in the paragraph “Specifying the Scope of a Command” of the Manipulating Objects section.

## Example

The following example is the object method for a button on a form that contains Area. When the object method executes, it checks to see that at least one object is selected; if so, it puts the names of the objects into the *\$Name* variable. It then tests to see if the names are all the same and displays an appropriate alert.

```

    If (DR Count (Area;0) > 0)
        `If at least one object is selected
⇒   $Name := DR Get name (Area;0)
        `Get the name(s)
    If (29 = DR Error)
        `Check if they were the same
        ALERT ("These objects don't have the same name!")
        `If not tell the user
    Else
        `If they are the same
        ALERT ("These objects are named "+$Name)
        `Tell the user the name
    End if
End if
```

## See Also

DR Get ID, DR SET NAME.

DR Get object type (area; scope) → Integer

Parameter	Type	Description
area	Longint	→ 4D Draw area
scope	Longint	→ -1=All 0=Selected >0=Object ID
Function result	Integer	← Type for the object(s) in area described by scope

### Description

The DR Get object type command returns the object type for the object(s) in area described by scope.

An object's type is described by an integer code and cannot be changed after an object is created.

- If scope equals -1, the function returns the object type for all objects in the document.
- If scope equals 0, the function returns the object type for the selected objects. In both cases, if the types of the objects are not equal, DR Get object type returns -32000, and DR Error returns error number 29.
- If scope exceeds 0, it must equal a specific object's ID; the function returns that object's type. If the object does not exist, DR Get object type returns -32000 and DR Error returns error number 2.

The following table lists all object codes:

Code	Object Type
1	Text object
3	PICT
4	Bitmap
5	Rectangle/Round rectangle
6	Polygon/Freehand object
7	Oval
8	Arc
9	Line
10	Group

## Example

The following example is the object method for a button on a form that contains area. When the object method executes, it checks to see that only rectangles are selected and then rounds their corners. If objects other than rectangles are selected, it alerts the user.

```
⇒   If (DR Get object type (Area;0) = 5)
      `Are selected objects rectangles?
      DR SET CORNER ROUNDING (Area;0;0.2)
      `Set their corner rounding
  Else
      `If other types are selected
      ALERT ("Only rectangles can be rounded!")
      `Alert the user
  End if
```

## See Also

DR Get ID.



---

DR GET POLYGON VERTEX (area; scope; number; vertexH; vertexV)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=First polygon in the document 0=Selected >0=Object ID
number	Integer	→	Number of vertex to edit
vertexH	Number	←	Horizontal position
vertexV	Number	←	Vertical position

### Description

The command DR GET POLYGON VERTEX returns into the vertexH and vertexV variables the position of the vertex specified by number for the polygon in area described by scope.

- If scope equals -1, the command returns coordinates for the first polygon in the document.
- If scope equals 0, the command returns coordinates for the first polygon in the current selection.
- If scope is greater than 0, it must be equal to a specific polygon's object ID; the command returns that polygon's coordinates. If the object does not exist, DR GET POLYGON VERTEX does nothing, and DR Error returns error number 2. If the object described by scope is not a polygon, DR GET POLYGON VERTEX returns -32000 for vertexH and vertexV, and DR Error returns error number 47.

number is the number of the vertex within the polygon. Vertices are numbered in the order in which they were created. If number exceeds the number of vertices in the polygon, DR GET POLYGON VERTEX does nothing, and DR Error returns error number 49.

vertexH and vertexV are the coordinates for the vertex, and are expressed in base units. Use the DR Base to scale function to convert from base units to scale units. vertexH and vertexV are returned as offsets from the current origin.

- Positive values indicate a position below or to the right of the origin.
- Negative values indicate a position above or to the left of the origin.

### Example

See the example for the DR SET POLYGON VERTEX command.

### See Also

DR SET POLYGON VERTEX.

DR Get refnum (area; scope) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Object ID
Function result	Longint	←	Reference number for the objects in area

### Description

The command DR Get refnum returns the reference number for the objects in area described by scope. A reference number is a long integer associated with an object and is not necessarily unique. Reference numbers can only be manipulated procedurally.

- If scope equals -2, the function returns the default reference number.
- If scope equals -1, the function returns the reference number for all objects in the document. If the reference numbers for the objects are not equal, DR Get refnum returns -32000, and DR Error returns error number 29.
- If scope equals 0, the function returns the reference number for the selected objects. If the reference numbers for the objects are not equal, DR Get refnum returns -32000, and DR Error returns error number 29.
- If scope is greater than 0, it must be equal to a specific object's ID; the function returns that object's reference number. If the object does not exist, DR Get refnum returns -32000, and DR Error returns error number 2.

### Example

The following example is the object method for a button on a form that contains area. When the object method executes it checks to see if only one object is selected, searches in the [Parts] table for the corresponding record and displays its description.

```

If (DR Count (Area;0) = 1)
⇒   QUERY ([Parts];[Parts]Refnum = DR Get refnum (Area;0))
    ALERT ("This object is a " + [Parts]Description)
End if

```

### See Also

DR Get ID, DR SET REFNUM.

---

DR Get rotation (area; scope) → Integer

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Object ID
Function result	Integer	←	Rotation in degrees of the objects in area

### Description

The command DR Get rotation returns the rotation in degrees of the objects in area described by scope and has a range of 0 to 359°.

Rotation degrees are given counter-clockwise from 0°.

- If scope equals -1, the function returns the rotation for all objects in the document.
- If scope equals 0, the function returns the rotation for the selected objects. If the rotation is not the same for all of the objects described by scope, DR Get rotation returns -32000, and DR Error returns error number 29.
- If scope is greater than 0, it must be equal to a specific object's ID; the function returns that object's rotation. If the object does not exist, DR Get rotation returns -32000, and DR Error returns error number 2.

### Example

The following example checks to see if any objects in Area are rotated and, if any are rotated, sets them back to 0°.

```
⇒  If (DR Get rotation (Area;-1) = -32000)
      DR ROTATE (Area;-1;0;0)
      End if
```

### See Also

DR Get ID, DR ROTATE.

---

DR Get text (area; scope) → Text

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-3=Selected characters -1=All 0=Selected >0=Object ID
Function result	Text	←	Text of the objects in area

### Description

The command DR Get text returns the text of the objects in area described by scope.

- If scope equals -3, the function returns the highlighted text of the selected object. If the insertion point is between characters, DR Get text returns an empty string.
- If scope equals -1, the function returns the text for the first text object in the document.
- If scope equals 0, the function returns the text of the first selected text object.
- If scope is greater than 0, it must be equal to a specific text object's ID; the function returns that object's text. If the object does not exist, DR Get text returns "\*\*\*\*\*" and DR Error returns error number 2.

### Example

See the example for DR SET HIGHLIGHT.

### See Also

DR SET TEXT.

---

DR GET TEXT ATTRIBUTES (area; scope; font; size; style; justification; frame; expansion)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-3=Selected characters -2=Default -1=All 0=Selected objects >0=Object ID
font	Integer	←	Font number
size	Integer	←	Font size in points
style	Integer	←	Font style
justification	Integer	←	Font justification
frame	Integer	←	0=Variable 1=Fixed
expansion	Integer	←	0=Down 1=Up

### Description

The command DR GET TEXT ATTRIBUTES returns into the integer variables font, size, style, justification, frame, and expansion the text attributes for the text object(s) in area described by scope.

- If scope equals -3, the command returns the text attributes for the highlighted text of the selected object. If the insertion point is between characters, DR GET TEXT ATTRIBUTES returns the attributes for the character to the left of the pointer.
- If scope equals -2, the command returns the default text attributes.
- If scope equals -1, the command returns the text attributes for all objects in the document.
- If scope equals 0, the command returns the text attributes for the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID; the command returns that object's attributes. If the object does not exist, DR GET TEXT ATTRIBUTES returns -32000 for each attribute, and DR Error returns error number 2.

font is the ID of the font in your system. This is the same value that is returned by the DR Font number function. You can use DR Font name to determine the name of the font (i.e., Arial, Chicago, and so on).

size is the size in points of the highlighted text or text objects.

style is a composite number that results from the addition of several style numbers. The following table lists style numbers:

Value	Style
0	Plain
1	Bold
2	Italic
4	Underline
8	Outlined
16	Shadowed

justification is the alignment of the text within the text block. The following table lists the possible values for justification:

Value	Justification
0	Left
1	Center
2	Right

If any of the attributes are not the same for all of the text objects described by scope, DR GET TEXT ATTRIBUTES returns -32000 for that attribute.

frame describes whether the text object has a fixed size.

- If frame equals 0, the height of the object is variable and is determined by the amount of text and the width of the text object.
- If frame equals 1, the height of the object is fixed and any text that does not fit within the text object is truncated.

expansion describes which direction text will flow within the object when it wraps to the next line.

- If expansion equals 0, the text object expands down the page to accommodate a new line of text.
- If expansion equals 1, the text object expands up the page when a line of text is added.

### Example

The following example gets the text attributes for the selected objects and then displays an alert that tells the user the font name.

```
⇒ DR GET TEXT ATTRIBUTES (Area;0;vFont;vSize;vStyle;vJust;vFrame;vFlow)
   If (vFont # -32000)
     ALERT ("You are using the font: " + DR Font name (vFont))
   End if
```

### See Also

DR SET TEXT ATTRIBUTES.

---

DR Get text width (area; scope) → Integer

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-3=Selected characters -1=All 0=Selection >0=Object ID
Function result	Integer	←	Width of the text

### Description

The command DR Get text width returns the width of the text in scope. The width is returned in the units specified for the document's ruler.

- If scope equals -3, the command returns the width of the selected characters.
- If scope equals -1, the command returns the longest text line in the 4D Draw area.
- If scope equals 0, the command returns the longest line in the current selection.
- If scope is greater than 0, the command returns the longest line in the object identified by the ID number passed as a parameter. If no object corresponds to this ID number, the command returns -32000 and DR Error returns 2.

The values for the scope parameter are listed in the paragraph “Specifying the Scope of a Command” of the Manipulating Objects section.

You can use the DR Get ruler function to determine the current ruler setup.





7

---

# DR Import and Export



---

DR ARRAY TO ATTRIBUTE (area; scope; attribute; array)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Group ID
attribute	Longint	→	Attribute code
array	Array	→	Source array

### Description

The command DR ARRAY TO ATTRIBUTE modifies attribute for each object in array, using a corresponding value. This command also applies to groups—each group is treated as a single object.

The object farthest in back in the stacking order corresponds to the first element in array.

- If scope equals -1, the command applies array to all objects in the document that are not in a group.
- If scope equals 0, the command applies array to the currently selected objects. A group is considered as a single object.
- If scope is greater than 0, it must be the ID for a group; the command applies array to the objects inside the group. This last syntax lets you modify objects in a group without ungrouping.

DR ARRAY TO ATTRIBUTE modifies only the objects that have corresponding elements in the array and will not create objects if there are more elements in the array than objects. To determine the value needed, see the command that affects that attribute. For example, to determine how to specify a fill pattern, see DR SET FILL ATTRIBUTES.

DR ARRAY TO ATTRIBUTE modifies information on only one “level” of objects. To modify specific objects inside of a group, pass the group’s ID as scope. Coordinate and size attributes are specified in base units. Use the DR ARRAY BASE TO SCALE command to convert an array of base units to scale units.

See Appendix A, Attribute Codes, for a complete list of attribute codes and corresponding array types. Note that several attributes cannot be modified.

## Example

The following example searches for a selection of records that contain colors (Longints), fills an array with the colors, and applies those to the selected objects, provided there are not more selected objects than colors.

```
    QUERY ([Colors];[Colors]Kind = "Standard")
    SELECTION TO ARRAY ([Colors]Color;aColor)
    If (DR Count (Area;0)<= Size of array (aColor))
⇒     DR ARRAY TO ATTRIBUTE (Area;0;13;aColor)
    Else
        ALERT("Too many objects selected!")
    End if
```

## See Also

DR ATTRIBUTE TO ARRAY, DR Get ID.

DR Array to polygon (area; arrayH; arrayV) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
arrayH	Numeric array	→	Array of horizontal values for vertices
arrayV	Numeric array	→	Array of vertical values for vertices
Function result	Longint	←	Object ID

### Description

The command DR Array to polygon creates a new polygon in area based on the arrays arrayH and arrayV and returns the new object's ID.

arrayH and arrayV describe the position of each polygon vertex. The two arrays can be of type Real, Longint, or Integer, and are specified in base units. Use the DR ARRAY SCALE TO BASE command to convert an array of scale units to base units. Each array must have at least three elements for the polygon to be successfully created. If the arrays are not the same size, the erroneous elements are ignored. To create a closed polygon, the last value in each array must match the first value.

### Examples

(1) The following example is the converse of the example used for DR POLYGON TO ARRAY. It fills two arrays from a selection of records and then, if the arrays have at least 3 elements, it creates a polygon using the arrays.

```

ARRAY REAL(ArrayH;0)
  `Declare the arrays
ARRAY REAL(ArrayV;0)
SELECTION TO ARRAY ([Polygon]FieldH;ArrayH;[Polygon]FieldV;ArrayV)
  `Fill the arrays
If (Size of array (ArrayH) >= 3)
  `If the arrays have enough elements
⇒   $New := DR Array to polygon (Area;ArrayH;ArrayV)
  `Create the polygon
End if

```

(2) The following example is the object method for a button on a form that contains Area. It asks the user for a number and then creates a polygon using the entered number of vertices.

```
$x := Num (Request ("How many vertices?"))
  `Ask how many vertices
If (OK = 1)
  `If they OK the request
  DR SET ORIGIN (Area;2;2;0)
  `Set the origin to 2, 2.
  ARRAY REAL (ArrayH;$x)
  `Declare the arrays
  ARRAY REAL (ArrayV;$x)
  For ($i;1;$x)
    `Fill the arrays
    ArrayH{$i} := Sin ($i)
    ArrayV{$i} := Cos ($i)
  End for
⇒ $New := DR Array to polygon (Area;ArrayH;ArrayV)
  `Creates a polygon
End if
```

#### See Also

DR GET POLYGON VERTEX, DR POLYGON TO ARRAY.

---

DR ATTRIBUTE TO ARRAY (area; scope; attribute; array)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Group ID
attribute	Longint	→	Attribute code
array	Array	→	Array to fill

### Description

The command DR ATTRIBUTE TO ARRAY returns into array the value of attribute for each object in area described by scope.

- If scope equals -1, the command returns attribute for each object in the document that is not in a group.
- If scope equals 0, the command returns attribute for each selected object that is not in a group. Groups appear as a single object.
- If scope is greater than 0, it must be the ID for a group; the command returns attribute for all objects inside the group. This syntax lets you get information about objects in a group without ungrouping. Nested groups can be traversed by using subsequent calls to DR ATTRIBUTE TO ARRAY. See the example for this command.

If attribute is not applicable for a specified object, DR ATTRIBUTE TO ARRAY returns -32000 or “\*\*\*\*\*” depending on the type of the array.

For example, if you fill an array with attribute 24 (corner rounding) and one of the objects is a line, the element that corresponds to the line will contain 32000. Likewise, if the object is a group and attribute is not the same for all objects inside of the group, DR ATTRIBUTE TO ARRAY returns 32000 or “\*\*\*\*\*”.

The values returned by DR ATTRIBUTE TO ARRAY are based on the attribute code used. To determine the meaning of the corresponding element, see the command affecting that attribute. For example, to determine how to interpret a fill pattern, see DR SET FILL ATTRIBUTES.

DR ATTRIBUTE TO ARRAY returns information on only one “level” of objects. For example, if scope is 0, and several of the selected objects are groups, each group is included in the resulting array as one element. You can then “look into” each group by passing its ID as the scope.

Coordinate and size attributes are returned in base units. Use the DR ARRAY BASE TO SCALE command to convert an array of base units to scale units.

See Appendix A, Attribute Codes for a complete list of attribute codes and corresponding array types.

## Example

The following example shows how to fill an array of object ID's and an array of object types for all objects in the document, regardless of whether the objects are in groups.

```
C_LONGINT ($i;$j)
  `Used as loop counters
ARRAY LONGINT (aID;0)
  `Will store the complete ID list
ARRAY LONGINT (aType;0)
  `Will store the complete Type list
ARRAY LONGINT (aGrpID;0)
  `Get the ID's inside of a group
ARRAY LONGINT (aGrpType;0)
  `Used to get Types inside of a group
$i := 0
  `Initialize loop counter
⇒ DR ATTRIBUTE TO ARRAY (Area;-1;0;aID)
  `Fill array with ID's of objects not in groups
⇒ DR ATTRIBUTE TO ARRAY(Area;-1;1;aType)
  `Fill array with types of objects not in groups
Repeat
  `Repeat until all groups have been "looked into"
  $i := $i + 1
  `Increment counter. Tells which object to work on.
  If(aType{$i} = 10)
    `If the object is a group
    ⇒ DR ATTRIBUTE TO ARRAY (Area;aID{$i};0;aGrpID)
      `Fill array with ID's of objects in group
    ⇒ DR ATTRIBUTE TO ARRAY (Area;aID{$i};1;aGrpType)
      `Fill array with Types of objects in group
    INSERT ELEMENT (aID;$i + 1;Size of array (aGrpID))
      `Insert into ID array, elements to hold this group
    INSERT ELEMENT(aType;$i + 1;Size of array (aGrpType))
      `Insert into Type array, elements to hold this group
    For($j;1;Size of array (aGrpID))
      `Loop through the group array...
      aID{$i + $j} := aGrpID{$j}
      `copy ID in group array into main array
      aType{$i + $j} := aGrpType{$j}
      `same for types
    End for
  End if
Until ($i = Size of array (aID))
  `Continue until objects are explored
```

## See Also

DR ARRAY TO ATTRIBUTE, DR Get ID.



---

DR POLYGON TO ARRAY (area; scope; arrayH; arrayV{; bezier})

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1 = Vertices of first object 0 = Vertices of selected object >0 = Object ID
arrayH	Numeric array	←	Array of horizontal values for vertices
arrayV	Numeric array	←	Array of vertical values for vertices
bezier	Numeric array	←	1=Bezier control point 0=Not a Bezier control point

### Description

The command DR POLYGON TO ARRAY returns into the arrays arrayH and arrayV, the position for each vertex of the polygon described by scope.

- If scope equals -1, the command returns the vertices of the first object in the document.
- If scope equals 0, the command returns the vertices of the first object selected.
- If scope is greater than 0, it must be equal to a specific polygon's ID; the command returns that polygon's vertices. If the object does not exist, DR POLYGON TO ARRAY returns empty arrays, and DR Error returns error number 2.

If the object described by scope is not a polygon, DR POLYGON TO ARRAY returns empty arrays and DR Error returns error number 47.

arrayH and arrayV are the horizontal and vertical positions for each vertex of the polygon; these are expressed in base units. Use the DR ARRAY BASE TO SCALE command to convert an array of base units to scale units. arrayH and arrayV can be arrays of type Real, Longint, or Integer, and must exist before calling DR POLYGON TO ARRAY. If you use one of the more restrictive array types (Longint or Integer), some information may be lost because values are truncated to whole numbers.

The bezier parameter lets you determine whether or not a control point is a Bezier control point. Bezier control points can only be created with the DR POLYGON CURVE command.

### Example

The following example fills two arrays with the vertices of the selected polygon. It then uses the 4th Dimension command `ARRAY TO SELECTION` to create records based on the values in the arrays.

```
    ARRAY REAL(ArrayH;0)
      `Declare the arrays
    ARRAY REAL(ArrayV;0)
⇒  DR POLYGON TO ARRAY (Area;0;ArrayH;ArrayV)
      `Get vertices
    If (DR Error=0)
      ARRAY TO SELECTION (ArrayH;[Polygon]FieldH;ArrayV;[Polygon]FieldV)
        `Create records
    End if
```

### See Also

DR Array to polygon, DR Get object type.

8

---

# DR Object Creation



---

DR Draw arc (area; start; length; radiusH; radiusV; centerH; centerV) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
start	Integer	→	starting angle in degrees
length	Integer	→	Length of Arc in degrees
radiusH	Number	→	Horizontal radius
radiusV	Number	→	Vertical radius
centerH	Number	→	Horizontal location of arc center
centerV	Number	→	Vertical location of arc center
Function result	Longint	←	Object ID

### Description

The command DR Draw arc creates a new arc in area and returns the object's ID. The new arc is created with the default line and fill attributes.

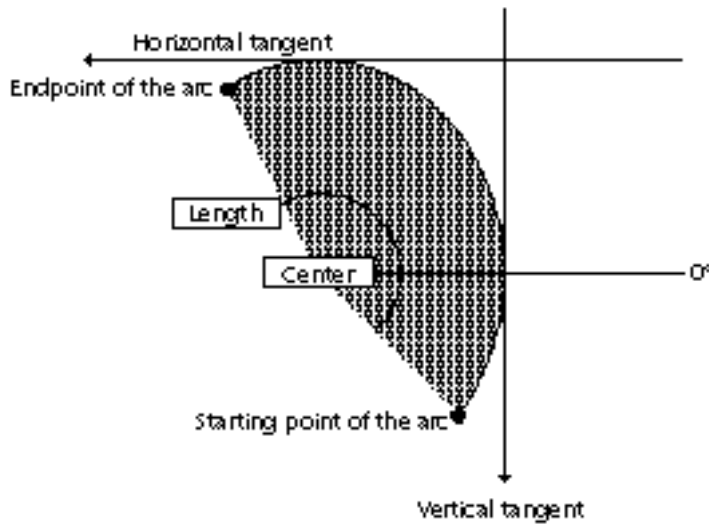
start is the beginning of the arc in degrees. start is the point that begins the arc, moving counter-clockwise from 0°. start must be less than 360°.

length is the length of the arc in degrees. The end of an arc is start + length % 360. Arcs have a maximum length of 359°.

radiusH and radiusV are the horizontal and vertical radii of the arc expressed in base units. As the basis of an arc is an oval, not necessarily a circle, these two numbers can be different. If radiusH equals radiusV, the arc is a portion of a circle.

centerH and centerV are the horizontal and vertical coordinates of the center of the arc with respect to the origin and are expressed in base units. Use the DR Base to scale function to convert from base units to scale units.

The following diagram illustrates the parameters that describe an arc:



### Example

The following example creates ninety 4° arcs. Each arc uses a different color but the same coordinates for the center. The effect is a circle filled with a rainbow gradient. This example requires a color monitor set to 8-bit depth.

```

For ($i ;0;89)
  DR SET FILL ATTRIBUTES (Area;-2;3;DR Index to color ($i + 17))
⇒   $ID := DR Draw arc (Area;$i * 4;4;2;2;2;2)
End for

```

### See Also

DR GET ARC SPECS, DR SET ARC SPECS.

DR Draw line (area; startH; startV; endH; endV; mode) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
startH	Number	→	Horizontal position of start
startV	Number	→	Vertical position of start
endH	Number	→	Horizontal position of end
endV	Number	→	Vertical position of end
mode	Integer	→	0=Absolute, 1=Relative
Function result	Longint	←	Object ID

### Description

The command DR Draw line creates a new line in area and returns the object's ID.

The new line is created with the default line attributes and positioned according to startH, startV, endH, and endV. All four coordinates are expressed in base units. Use the DR Scale to base function to convert from scale units to base units.

startH and startV are specified as offsets from the current origin.

endH and endV can be specified as offsets from the current origin (absolute) or offsets from startH and startV (relative).

If mode equals 0, endH and endV are absolute coordinates. If mode equals 1, endH and endV are relative coordinates. Positive values indicate a direction down or right. Negative values indicate a direction up or left.

### Examples

(1) The following example uses DR Draw line with absolute coordinates. It sets the origin to the upper left corner of the document and then draws a line that starts one in/one down and ends two in/two down.

```
DR SET ORIGIN (Area;0;0;0)
⇒ $ID := DR Draw line (Area;1;1;2;0)
```

(2) The following example uses DR Draw line with relative coordinates. It draws a line that starts at the current origin and ends three base units to the right.

```
⇒ $ID := DR Draw line (Area;0;0;3;0;1)
```

### See Also

DR SET ENDMARKS, DR SET LINE ATTRIBUTES.

DR Draw oval (area; startH; startV; endH; endV; mode) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
startH	Number	→	Horizontal position of start
startV	Number	→	Vertical position of start
endH	Number	→	Horizontal position of end
endV	Number	→	Vertical position of end
mode	Integer	→	0=Absolute 1=Relative
Function result	Longint	←	Object ID

### Description

The command DR Draw oval creates a new oval in area and returns the object's ID.

The new oval is created with the default line and fill attributes and positioned according to startH, startV, endH, and endV. All four coordinates are expressed in base units. Use the DR Scale to base function to convert from scale units to base units.

startH and startV are specified as offsets from the current origin.

endH and endV can be specified as offsets from the current origin (absolute) or offsets from startH and startV (relative).

If mode equals 0, endH and endV are absolute coordinates. If mode equals 1, endH and endV are relative coordinates. Positive values indicate a direction down or right. Negative values indicate a direction up or left.

### Example

The following example creates a one unit circle with its center at the current origin and then changes the circle's pattern.

```
⇒ $ID := DR Draw oval (Area;-0.5;-0.5;0.5;0.5;0)
   DR SET FILL ATTRIBUTES (Area;$ID;3;0)
```

### See Also

DR SET FILL ATTRIBUTES, DR SET LINE ATTRIBUTES.



---

DR Draw rectangle (area; startH; startV; endH; endV; mode; round) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
startH	Number	→	Horizontal position of start
startV	Number	→	Vertical position of start
endH	Number	→	Horizontal position of end
endV	Number	→	Vertical position of end
mode	Integer	→	0 = Absolute 1 = Relative
round	Number	→	Corner rounding amount
Function result	Longint	←	Object ID

### Description

The command DR Draw rectangle creates a new rectangle in area and returns the object's ID.

The new rectangle is created with the default line and fill attributes and positioned according to startH, startV, endH, and endV. All four coordinates are expressed in base units. Use the DR Scale to base function to convert from scale units to base units.

startH and startV are specified as offsets from the current origin.

endH and endV can be specified as offsets from the current origin (absolute) or offsets from startH and startV (relative).

- If mode equals 0, endH and endV are absolute coordinates.
- If mode equals 1, endH and endV are relative coordinates. Positive values indicate a direction down or right. Negative values indicate a direction up or left.

round controls the amount of corner rounding of the new rectangle. round is specified in base units. If round equals 0, the rectangle is created with no corner rounding.

## Example

The following example creates a 10 by 10 grid of rectangles with random patterns and colors. Updating is left on to view the progress of the method.

```
DR SET ORIGIN (Area;0;0;0)
  `Set origin to upper left of the document
For ($i;0;9)
  ` $i is the horizontal position for each rectangle
  For ($j;0;9)
  ` $j is the vertical position for each rectangle
  DR SET FILL ATTRIBUTES (Area;-2;Random % 33 + 3;Random ^ 2)
  `Set default fill attributes to random color & pattern
⇒   $ID := DR Draw rectangle (Area;$i;$j;1;1;1;0)
  `Draw a new rectangle. Note relative coordinates.
  End for
End for
```

## See Also

DR SET FILL ATTRIBUTES, DR SET LINE ATTRIBUTES.

---

DR Draw text (area; startH; startV; endH; endV; text{; frame{; expansion})) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
startH	Number	→	Horizontal position of start
startV	Number	→	Vertical position of start
endH	Number	→	Horizontal position of end
endV	Number	→	Vertical position of end
text	Text	→	Text of new text object
frame	Integer	→	0=Variable, 1=Fixed
expansion	Integer	→	0=Down, 1=Up
Function result	Longint	←	Object ID

### Description

The command DR Draw text creates a new text object in area and returns the ID of the object. The new object is positioned according to startH, startV, endH, and endV and filled with text. All four coordinates are expressed in base units from the current origin. Use the DR Scale to base function to convert from scale units to base units.

text must be at least one character in length, otherwise the object is not created. The font, size, style, and justification of the new text object is determined by the current default text attributes. See the DR SET TEXT ATTRIBUTES command.

The optional frame parameter controls whether the text object has a fixed size.

- If frame equals 0 or is not specified, the height of the object is variable and is determined by the amount of text and the width of the text object.
- If frame equals 1, the height of the object is fixed and any text that does not fit within the text object is truncated.
- If frame is omitted, expansion must also be omitted.

expansion describes the direction in which text will flow within the object when it wraps to the next line.

- If expansion equals 0, the text object expands down the page to accommodate a new line of text.
- If expansion equals 1, the text object expands up the page when a line of text is added.

frame and expansion are especially useful when working with text objects that contain references. Since references can insert any amount of text, having control over the size and flow direction of text objects during printing is very useful.

### See Also

DR GET TEXT ATTRIBUTES, DR SET TEXT ATTRIBUTES.

---

DR End polygon (area) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
Function result	Longint	←	Object ID

### Description

The command DR End polygon completes the polygon in area and returns the object's ID. The new polygon is created with the default line and fill attributes.

### Example

The following example creates a complex polygon centered two units down and two units to the right of the upper left corner of the page.

```

DR START POLYGON (Area)
DR SET ORIGIN (Area;2;2;0)
For ($i;1;51)
  DR POLYGON LINE (Area;Sin($i);Cos($i);0)
End for
⇒ $ID := DR End polygon (Area)

```

### See Also

DR POLYGON LINE, DR START POLYGON.

---

DR Objects to bitmap (area; scope; option) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All objects 0=Selected objects >0=ID
option	Integer	→	Option for converting the objects in scope (0, 1, or 2)
Function result	Longint	←	Object ID

### Description

The command DR Objects to bitmap changes the objects specified by area and scope to a bitmap.

- If option equals 0, the function converts objects in scope to a black and white bitmap.
- If option equals 1, the function converts objects in scope to a color bitmap with a transparent background.
- If option equals 2, the function converts objects in scope to a color bitmap with a white background.

In both cases, the function returns the ID number of the object of type Picture created by DR Objects to bitmap.

The color bitmap uses the number of colors selected in the Monitor control panel. The color bitmap is automatically converted to an object of type Picture; so, if a different number of colors is chosen in the Monitor control panel, the bitmap image will not be affected.

**Note:** Converting objects to a bitmapped image is irreversible. The bitmapped image loses all of its former attributes, including its name.

### See Also

DR ADD TO BITMAP.

---

DR PLACE PICTURE (area; picture; location)

Parameter	Type		Description
area	Longint	→	4D Draw area
picture	Picture	→	4D Picture to place
location	Integer	→	0=Normal 1=Center 2=Origin

### Description

The command DR PLACE PICTURE pastes picture into area at a point determined by location.

picture must be a valid 4th Dimension picture expression.

- If location equals 0, the command pastes picture as if from the Clipboard, that is, it places picture where the mouse was last clicked in area.
- If location equals 1, the command places picture in the center of the window that displays area.
- If location equals 2, the command places picture at the origin.

### Example

The following example pastes the contents of the Picture field [Logos]Logo into Area for the specified company.

```

MyRequest:=Request ("Which company's logo do you want?")
If (OK=1)
  QUERY ([Logos];[Logos]Company=MyRequest)
  If (Records in selection([Logos]>0)
⇒    DR PLACE PICTURE(Area;[Logos]Logo;2)
  Else
    ALERT("This company does not exist.")
  End if
End if

```

### See Also

DR Area to picture, DR Clipboard to picture, DR SET ORIGIN.

---

DR POLYGON CURVE (area; x1; y1; x2; y2; targetX; targetY; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
x1	Number	→	X coordinate of control point 1
y1	Number	→	Y coordinate of control point 1
x2	Number	→	X coordinate of control point 2
y2	Number	→	Y coordinate of control point 2
targetX	Number	→	X coordinate of ending point
targetY	Number	→	Y coordinate of ending point
mode	Integer	→	0 = Absolute 1 = Relative

### Description

The command DR POLYGON CURVE allows you to draw a curve.

A line begins from the starting point as if it were heading for Control Point 1 (x1, y1).

A second line begins from the ending point and heads in the direction of Control Point 2 (x2,y2). The lines are curved so that they will intersect halfway between the starting point and the ending point.

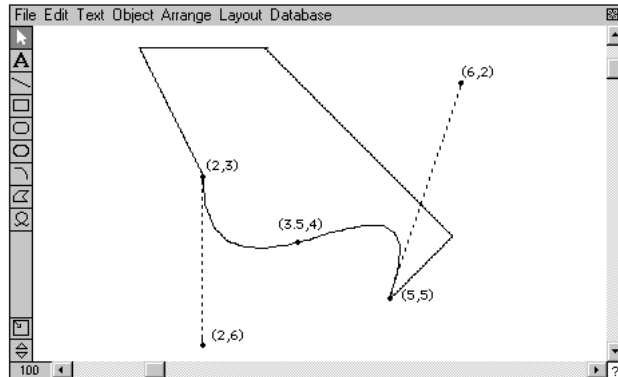
### Example

The following method creates a curved polygon using several straight lines and one curved line:

```

DR START POLYGON (MyArea)
DR POLYGON LINE (MyArea;1;1;0)
DR POLYGON LINE (MyArea;3;1;0)
DR POLYGON LINE (MyArea;6;4;0)
DR POLYGON LINE (MyArea;5;5;0)
⇒ DR POLYGON CURVE (MyArea;6;2;2;6;2;3;0)
DR POLYGON LINE (MyArea;1;1;0)
$PolyID := DR End polygon (MyArea)

```



Point (3.5,4) is halfway between points (2,3) and (5,5), and is the point where the two curved lines meet.

#### See Also

DR End polygon, DR POLYGON LINE, DR START POLYGON.



---

DR POLYGON LINE (area; vertexH; vertexV; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
vertexH	Number	→	Horizontal position
vertexV	Number	→	Vertical position
mode	Integer	→	0=Absolute, 1=Relative

### Description

The command DR POLYGON LINE adds a vertex to the polygon that is currently being built for area.

vertexH and vertexV can be specified as offsets from the current origin (absolute) or offsets from the previous vertex (relative). Both vertexH and vertexV are expressed in base units. Use the function DR Scale to base to convert from scale units to base units.

- If mode equals 0, vertexH and vertexV are absolute coordinates.
- If mode equals 1, vertexH and vertexV are relative coordinates.

The first vertex in a polygon is always absolute. Positive values indicate a direction down or right. Negative values indicate a direction up or left.

To make a closed polygon, the last vertex must match the first. A polygon requires at least three vertices.

To add a new vertex, the drawing process must already have been started by a call to DR START POLYGON. If you added several vertices to a polygon and then make another call to DR START POLYGON, the vertices are erased and the polygon is started again. Each call to DR POLYGON LINE is equivalent to a user mouse-click when a new polygon is being drawn.

### Example

The following example creates a diamond-shaped polygon using relative references.

```

DR START POLYGON (Area)
⇒ DR POLYGON LINE (Area;1.5;1;0)
⇒ DR POLYGON LINE (Area;0.5;0.5;1)
⇒ DR POLYGON LINE (Area;-0.5;0.5;1)
⇒ DR POLYGON LINE (Area;-0.5;-0.5;1)
⇒ DR POLYGON LINE (Area;0.5;-0.5;1)
$ID := DR End polygon (Area)

```

### See Also

DR End polygon, DR POLYGON CURVE, DR START POLYGON.

---

DR START POLYGON (area)

Parameter	Type		Description
area	Longint	→	4D Draw area

### Description

The command DR START POLYGON begins a new polygon for area. This command tells 4D Draw to create a space in memory for building the new polygon. The polygon is incomplete, and is therefore not drawn, until a matching call to DR End polygon is made. If a polygon is already in progress, DR START POLYGON starts the drawing process again.

### Example

The following example creates diamond-shaped polygon.

```
⇒  DR START POLYGON (Area)
    DR POLYGON LINE (Area;1.5;1;0)
    DR POLYGON LINE (Area;2;1.5;0)
    DR POLYGON LINE (Area;1.5;2;0)
    DR POLYGON LINE (Area;1;1.5;0)
    DR POLYGON LINE (Area;1.5;1;0)
    $ID := DR End polygon (Area)
```

### See Also

DR End polygon, DR POLYGON CURVE, DR POLYGON LINE.

9

---

# DR Object Manipulation



DR ADD TO BITMAP (area; scope; objectID)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All objects 0=Selected objects >0=ID
objectID	Longint	→	ID number of the bitmapped image

### Description

The command DR ADD TO BITMAP adds the objects in scope to a bitmapped image identified by objectID.

Converting objects to a bitmapped image is irreversible. The bitmapped image loses all of its former attributes, including its name.

### See Also

DR Objects to bitmap.

DR ALIGN (area; scope; horizontal; vertical)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected
horizontal	Integer	→	0=None 1=Left 2=Middle 3=Right
vertical	Integer	→	0=None 1=Top 2=Middle 3=Bottom

### Description

The command DR ALIGN aligns the objects in area described by scope.

- If scope equals -1, DR ALIGN aligns all objects in the document.
- If scope equals 0, DR ALIGN aligns the selected objects.

The objects described by scope are aligned according to the horizontal and vertical parameters.

Following are the values for the horizontal parameter, a description of their effects, and the corresponding icons in the Align Objects dialog box:

0 No horizontal alignment

1 Left sides



2 Centers



3 Right sides



- If horizontal equals 0, DR ALIGN does no horizontal alignment.
- If horizontal equals 1, the command aligns the left sides.
- If horizontal equals 2, the command aligns the horizontal centers.
- If horizontal equals 3, the command aligns the right sides.

DR ALIGN (area; scope; horizontal; vertical)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected
horizontal	Integer	→	0=None 1=Left 2=Middle 3=Right
vertical	Integer	→	0=None 1=Top 2=Middle 3=Bottom

### Description

The command DR ALIGN aligns the objects in area described by scope.

- If scope equals -1, DR ALIGN aligns all objects in the document.
- If scope equals 0, DR ALIGN aligns the selected objects.

The objects described by scope are aligned according to the horizontal and vertical parameters.

Following are the values for the horizontal parameter, a description of their effects, and the corresponding icons in the Align Objects dialog box:

0 No horizontal alignment

1 Left sides



2 Centers



3 Right sides



- If horizontal equals 0, DR ALIGN does no horizontal alignment.
- If horizontal equals 1, the command aligns the left sides.
- If horizontal equals 2, the command aligns the horizontal centers.
- If horizontal equals 3, the command aligns the right sides.

Following are the values for the vertical parameter, a description of their effects, and the corresponding icons in the Align Objects dialog box:

0 No vertical alignment

1 Tops



2 Centers



3 Bottoms



- If vertical equals 0, DR ALIGN does no vertical alignment.
- If vertical equals 1, the command aligns the tops.
- If vertical equals 2, the command aligns the vertical centers.
- If vertical equals 3, the command aligns the bottoms.

### Example

The following example centers the selected objects both horizontally and vertically.

⇒ *DR ALIGN* (Area;0;2;2)

### See Also

DR Get ID.



---

DR Count (area; scope) → Integer

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Group ID
Function result	Integer	←	Number of objects in area specified by scope

### Description

The command DR Count returns the number of objects in area specified by scope.

- If scope equals -1, DR Count returns the number of objects in the document that are not in a group. Groups appear as a single object.
- If scope equals 0, DR Count returns the number of objects currently selected that are not in a group. Groups appear as a single object.
- If scope is greater than 0, it must be the ID for a group; DR Count returns the number of objects inside the group. This last syntax lets you get information about objects in a group without ungrouping. You can examine nested groups by using subsequent calls to DR Count.

### Example

The following example opens an alert that displays the number of currently selected objects.

```
⇒ $Temp := DR Count (Area;0)
   ALERT ("You have selected " + String ($Temp) + " object(s).")
```

### See Also

DR Get ID.

DR DELETE (area; scope)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Object ID

### Description

The command DR DELETE deletes the objects in area described by scope.

- If scope equals -1, DR DELETE deletes all objects in the document.
- If scope equals 0, DR DELETE deletes the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID; the command deletes that object. If the object does not exist, DR Error returns error number 2.

### Example

The following example deletes the object in Area whose ID is 5.

⇒ *DR DELETE* (Area;5)

### See Also

DR Get ID.

---

DR GROUP (area; scope)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected

### Description

The command DR GROUP groups the objects in area described by scope.

- If scope equals -1, DR GROUP groups all objects in the document.
- If scope equals 0, DR GROUP groups the selected objects. When objects are grouped, a new object of type Group is created.

After a call to DR GROUP, the command automatically selects the new object; you can use the DR Get ID function to obtain its ID. If less than two objects are described by scope, DR GROUP does nothing.

### Example

The following example checks to see if more than one object is selected and, if so, asks the user to supply a name. It then groups the selected objects and gives the new group the user supplied name.

```

    If (DR Count (Area;0) > 1)
      $Temp := Request ("Please name this new group.")
      If (OK = 1)
⇒      DR GROUP (Area;0)
        DR SET NAME (Area;0;$Temp)
      End if
    Else
      ALERT ("Not enough objects selected.")
    End if

```

### See Also

DR Get ID, DR UNGROUP.

DR HIDE (area; scope; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All objects 0=Selected objects >0=ID
mode	Integer	→	1=Hide 0=Show

### Description

The command DR HIDE hides or shows the objects in scope. This command performs the same function as the **Hide** and **Show All** menu items do for a selection.

If scope equals 0 and mode equals 0, all hidden objects remain selected. Deselect the objects before returning control to the user or applying another command.

---

DR LOCK (area; scope; code; action)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Object ID
code	Longint	→	Attributes to affect
action	Integer	→	0=Unlock 1=Lock 2=Toggle

### Description

The command DR LOCK locks or unlocks the attributes described by code for the objects in area described by scope.

- If scope equals -1, DR LOCK affects all objects in the document.
- If scope equals 0, DR LOCK affects the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID and that object's attributes are affected. If the object does not exist, DR Error returns error number 2.

The objects described by scope are locked or unlocked according to the code and action parameters. code describes the attributes to act upon and is expressed as the sum of lock codes.

The following is the list of lock codes:

Value	Lock
-1	All
1	Deletion
2	Ungroup
8	Name
32	Size
128	Location
2048	Rotation
4096	Fill pattern
8192	Fill color
16384	Line pattern
32768	Line color
65536	Line width
131072	Endmarks
524288	Text font
1048576	Text size
2097152	Text style
4194304	Text justification
8388608	Text editing
16777216	Corner rounding
33554432	Shape

action specifies what to do to the attributes described by code.

- If action equals 0, DR LOCK unlocks the attributes described by code.
- If action equals 1, DR LOCK locks the attributes described by code.
- If action equals 2, DR LOCK toggles the current state of the attributes, that is, it unlocks locked attributes and locks unlocked ones. Attributes not specified by code remain unchanged.

### Example

The following example unlocks all attributes for the selected objects in Area.

⇒ *DR LOCK* (Area;0;-1;0)

### See Also

DR Get ID.

DR MOVE (area; scope; moveH; moveV; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All, 0=Selected, >0=Object ID
moveH	Number	→	Horizontal movement
moveV	Number	→	Vertical movement
mode	Integer	→	0=Absolute 1=Relative

### Description

The command DR MOVE repositions the objects in area described by scope.

- If scope equals -1, DR MOVE repositions all objects in the document.
- If scope equals 0, DR MOVE repositions the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID; DR MOVE repositions that object. If the object does not exist, DR Error returns error number 2.

The objects described by scope are moved according to the moveH, moveV, and mode parameters. You can specify moveH and moveV as offsets from the current origin (absolute) or as offsets from the object's current position (relative).

- If mode equals 0, moveH and moveV are absolute coordinates.
  - If mode equals 1, moveH and moveV are relative coordinates.
- Positive values indicate a direction down or right. Negative values indicate a direction up or left. Both moveH and moveV are expressed in base units. Use the DR Scale to base function to convert from scale units to base units.

### Example

The following examples are the object methods for four different buttons named bLeft, bRight, bUp, and bDown. Each button moves the selected objects one base unit in the given direction.

```
⇒ DR MOVE(Area;0;-1;0;1) `bLeft
⇒ DR MOVE(Area;0;1;0;1) `bRight
⇒ DR MOVE(Area;0;0;-1;1) `bUp
⇒ DR MOVE(Area;0;0;1;1) `bDown
```

### See Also

DR Get ID, DR SCALE, DR SIZE.

---

DR ROTATE (area; scope; amount; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Object ID
amount	Integer	→	Amount of rotation in degrees
mode	Integer	→	0=Absolute 1=Relative

### Description

The command DR ROTATE rotates the objects in area described by scope.

- If scope equals -1, DR ROTATE rotates all objects in the document.
- If scope equals 0, DR ROTATE rotates the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID and that object is rotated. If the object does not exist, DR Error returns error number 2.

The objects described by scope are rotated according to the amount and mode parameters.

You can specify amount as absolute or relative. If mode equals 0, amount is absolute and DR ROTATE rotates the object to an angle equal to amount degrees. If mode equals 1, amount is relative and DR ROTATE increases or decreases the object's rotation by amount.

A positive value for amount indicates a counter-clockwise rotation. A negative value for amount indicates clockwise rotation.

amount must be in the range  $\pm 359$ , or DR ROTATE does nothing, and DR Error returns error number 15.

### Example

The following example is the object method for a button. It rotates all objects in the drawing back to 0°.

⇒ *DR ROTATE* (Area;-1;0;0)

### See Also

DR Get ID, DR Get rotation.



---

DR SCALE (area; scope; anchorH; anchorV; amountH; amountV)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Object ID
anchorH	Integer	→	0=None 1=Left 2=Middle 3=Right
anchorV	Integer	→	0=None 1=Top 2=Middle 3=Bottom
amountH	Number	→	Horizontal multiplier
amountV	Number	→	Vertical multiplier

### Description

The command DR SCALE scales the objects described by scope in area.

- If scope equals -1, DR SCALE scales all objects in the document.
- If scope equals 0, DR SCALE scales the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID; DR SCALE scales that object. If the object does not exist, DR Error returns error number 2.

anchorH and anchorV indicate which side of an object is anchored when it is scaled.

- If anchorH equals 0, the width of the object does not change and DR SCALE ignores any value for amountH.
- If anchorH equals 1, DR SCALE anchors the left side of the object.
- If anchorH equals 2, the command anchors the center of the object.
- If anchorH equals 3, the command anchors the right side of the object.
- As with anchorH, if anchorV equals 0, the height of the object does not change and DR SCALE ignores any value for amountV.
- If anchorV equals 1, DR SCALE anchors the top of the object.
- If anchorV equals 2, the command anchors the center of the object.
- If anchorV equals 3, the command anchors the bottom of the object.

DR SCALE scales objects described by scope according to the amountH and amountV parameters.

- If amountH and amountV are greater than 1, DR SCALE increases object size.
- If amountH and amountV are less than 1, DR SCALE decreases object size.

The new width of the object is equal to its current width, multiplied by amountH. The new height is similarly derived, using amountV.

### Example

The following examples are object methods for two different buttons named bDouble and bHalf. The first button doubles the size of the selected object, and the second cuts the object's size in half.

⇒ *DR SCALE* (Area;0;2;2;2;2) `bDouble  
*DR SCALE* (Area;0;2;2;0.5;0.5) `bHalf

### See Also

DR Get ID, DR MOVE, DR SIZE.

---

DR SIZE (area; scope; anchorH; anchorV; width; height; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Object ID
anchorH	Integer	→	0=None 1=Left 2=Middle 3=Right
anchorV	Integer	→	0=None 1=Top 2=Middle 3=Bottom
width	Number	→	Width
height	Number	→	Height
mode	Number	→	0=Absolute 1=Relative

### Description

The command DR SIZE resizes the objects described in area by scope.

- If scope equals -1, the command resizes all objects in the document.
- If scope equals 0, the command resizes the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID; the command resizes that object. If the object does not exist, DR Error returns error number 2.

anchorH and anchorV indicate which side of an object is anchored when it is resized.

- If anchorH equals 0, the width of the object does not change and DR SIZE ignores any value for width.
- If anchorH equals 1, the command anchors the left side of the object.
- If anchorH equals 2, the command anchors the center of the object.
- If anchorH equals 3, the command anchors the right side of the object.
- As with anchorH, if anchorV equals 0, the height of the object does not change and DR SIZE ignores any value for height.
- If anchorV equals 1, DR SIZE anchors the top of the object.
- If anchorV equals 2, the command anchors the center of the object.
- If anchorV equals 3, the command anchors the bottom of the object.

DR SIZE resizes the objects described by scope according to the width, height, and mode parameters. You can specify width and height as absolute or relative.

- If mode equals 0, width and height are absolute and DR SIZE makes the object width wide and height tall.
- If mode equals 1, width and height are relative and the command increases or decrease object size by width and height. Positive values indicate a direction down or right. Negative values indicate a direction up or left. Both width and height are expressed in base units. Use the DR Scale to base function to convert from scale units to base units.

### Example

The following example adds one unit of height and width to the object with ID equal to 5 using the center of the object as the anchor.

⇒ *DR SIZE* (Area;5;2;2;1;1;1)

### See Also

DR Get ID, DR MOVE, DR SCALE.

---

DR UNGROUP (area; scope; level)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Group ID
level	Integer	→	Number of levels to ungroup

### Description

The command DR UNGROUP ungroups the objects in area described by scope.

- If scope equals -1, DR UNGROUP ungroups all objects in the document.
- If scope equals 0, DR UNGROUP ungroups the selected objects.
- If scope is greater than 0, it must be equal to a specific group's ID and that object is ungrouped. If the object does not exist, DR Error returns error number 2.

level controls how many layers of grouping are removed.

- If level equals 1, DR UNGROUP ungroups only first level groups.
- If level equals 2, the command ungroups first level groups and any groups within those objects.
- If level equals -1, the command ungroups all levels of groups.

Choosing Ungroup from the Arrange menu is equivalent to ungrouping one level.

### Example

The following example ungroups all objects at all levels in Area.

⇒ *DR UNGROUP* (Area;-1;-1)

### See Also

DR Get ID, DR GROUP.



# 10

---

## DR Object Selection





---

DR SELECT (area; scope; action)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1=All 0=Selected >0=Object ID
action	Integer	→	0=Deselect 1=Select 2=Toggle

### Description

The command DR SELECT selects or deselects the objects in area described by scope.

- If scope equals -1, DR SELECT affects all objects in the document.
- If scope equals 0, DR SELECT affects the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID; DR SELECT affects that object. If the object does not exist, DR SELECT does nothing, and DR Error returns error number 2.

DR SELECT selects or deselects the objects described by scope are according to the action parameter.

- If action equals 0, the command deselects the objects described by scope.
- If action equals 1, the command deselects the objects described by scope.
- If action equals 2, the command toggles the current state of the objects, that is, it selects deselected objects and vice-versa.

Objects outside of scope are not affected by DR SELECT. That is, objects that are already selected in area and have not been specified by scope remain selected.

### Example

The following example deselects all objects in the document and then selects the object with an ID number equal to 1.

⇒ *DR SELECT* (Area;-1;0)

⇒ *DR SELECT* (Area;1;1)

### See Also

DR Get ID, DR SELECT BY ATTRIBUTE, DR SELECT BY REGION, DR SET HIGHLIGHT.

---

DR SELECT BY ATTRIBUTE (area; conjunction; code; value)

Parameter	Type		Description
area	Longint	→	4D Draw area
conjunction	Integer	→	0=Only 1=Union 2=Intersection 3=Difference
code	Integer	→	Attribute code
value	Text	→	Value to compare to attribute

### Description

The command DR SELECT BY ATTRIBUTE selects or deselects objects in area based on the object's attributes. DR SELECT BY ATTRIBUTE compares value to the attribute described by code for each object and then selects or deselects those objects according to conjunction.

- If conjunction equals 0, the command selects only those objects that meet the criteria.
- If conjunction equals 1, the command selects those objects that meet the criteria along with any previously selected objects.
- If conjunction equals 2, the command selects those objects that meet the criteria among the currently selected objects.
- If conjunction equals 3, the command deselects the objects that meet the criteria among the currently selected objects.

DR SELECT BY ATTRIBUTE is similar to a built search in 4th Dimension. It allows you to combine search criteria using repetitive calls.

value is of type Text, even though most of the attributes are expressed numerically. This is because one of the attributes that can be searched is name, which must be expressed as a string. For all other attributes, you must supply a text representation of the number (i.e., "2" or "4.375") for value. To determine the value needed, see the command that affects that attribute. For example, to determine how to specify a fill pattern, see the DR SET FILL ATTRIBUTES command.

code is an integer from the standard list of attribute codes. See Appendix A, Attribute Codes for a complete list of attribute codes and corresponding array types. Note that you can search only some of the attributes.

DR SELECT BY ATTRIBUTE cannot select by object ID. Use DR SELECT to select objects by ID.

## Example

The following example selects all rectangles and ovals that are filled with pattern 3 (solid), which are not rotated (angle equals 0°), and which have a name (Name # "").

```
DR SET UPDATE MODE (Area;0)
  ^Turn updating off
⇒ DR SELECT BY ATTRIBUTE (Area;0;1;"5")
  ^Find rectangles
⇒ DR SELECT BY ATTRIBUTE (Area;1;1;"7")
  ^and ovals
⇒ DR SELECT BY ATTRIBUTE (Area;2;12;"3")
  ^that are filled with a solid pattern
⇒ DR SELECT BY ATTRIBUTE (Area;2;11;"0")
  ^and not rotated
⇒ DR SELECT BY ATTRIBUTE (Area;3;3;"")
  ^and not named with empty string
DR SET UPDATE MODE (Area;1)
  ^Turn updating on
```

## See Also

DR SELECT.

---

DR SELECT BY REGION (area; method; action; left; top; right; bottom)

Parameter	Type		Description
area	Longint	→	4D Draw area
method	Integer	→	0=Enclose, 1=Intersect
action	Integer	→	0=Deselect, 1=Select, 2=Toggle
left	Number	→	Left boundary
top	Number	→	Top boundary
right	Number	→	Right boundary
bottom	Number	→	Bottom boundary

### Description

The command DR SELECT BY REGION selects or deselects objects within a rectangular region of area. This command is the procedural equivalent of the user dragging a selection rectangle to select objects.

The region is described by left, top, right, and bottom. All four coordinates are specified in base units according to the current origin. Use the DR Scale to base function to convert from scale units to base units.

method determines whether objects must be completely enclosed to be affected.

- If method equals 0, objects must be completely enclosed within the region to be affected.
- If method equals 1, objects need only intersect the region to be affected.

The objects are selected or deselected based on the action parameter.

- If action equals 0, DR SELECT BY REGION deselects the objects.
- If action equals 1, the command selects the objects.
- If action equals 2, the command toggles the current state of the objects, that is, it deselects selected objects, and vice-versa.

DR SELECT BY REGION does not affect objects outside of the region. This means that if objects are already selected in area and are not within the region, they remain selected.

### Example

The following example deselects all objects in Area and then selects any objects on the first page of the document.

```
DR SELECT (Area;-1;0)
DR SET ORIGIN (Area;0;0;0)
⇒ DR SELECT BY REGION (Area;0;1;0;0;7.67;10.14)
```

### See Also

DR GET BOUNDARY.

11

---

# DR Printing



---

DR PRINT (area; message{; printDialog})

Parameter	Type		Description
area	Longint	→	4D Draw area
message	Integer	→	0 = Message off, 1 = Message on
printDialog	Integer	→	0 = Without dialog box, 1 = With dialog box

### Description

The command DR PRINT prints the document in area. Calling DR PRINT performs a function similar to choosing Print from the File menu, except that the Print Setup dialog box is not presented to the user. To display the Print Setup dialog box before printing, use DR DO COMMAND.

- If message equals 1, 4D Draw displays a dialog box that allows the user to cancel printing. If the user cancels printing, DR Error returns error number 55.
- If message equals 0, the dialog box is not displayed and the user cannot cancel printing.
- If the optional printDialog parameter equals 0, the standard Print File dialog box does not appear and the print job begins immediately.
- If printDialog equals 1, the standard Print File dialog box appears.

### Example

The following example creates an offscreen area, draws some objects, and prints the area:

```

$Area := DR New offscreen area
...
`Do some drawing here
⇒ DR PRINT($Area;0)
  `Print with messages off
DR DELETE OFFSCREEN AREA($Area)
  `Always delete offscreen areas when
  `you are done with them

```

### See Also

DR DO COMMAND, DR SET DOCUMENT SIZE.

---

DR PRINT BACKGROUND (area; message; printDialog)

Parameter	Type		Description
area	Longint	→	4D Draw area
message	Integer	→	0 = Message off, 1 = Message on
printDialog	Integer	→	0 = Without dialog box, 1 = With dialog box

### Description

The command DR PRINT BACKGROUND allows you to print only the objects on the background of area.

- If message equals 1, a dialog box appears, allowing the user to cancel the printing in progress. If the user cancels the print job, an error 55 is returned by DR Error.
- If message equals 0, this dialog box does not appear, and the user cannot cancel the print job.

printDialog determines whether or not the standard Print dialog box will appear.

- If printDialog equals 0, the standard Print File dialog box does not appear, and the print job begins immediately.
- If printDialog equals 1, the standard Print File dialog box appears.

### See Also

DR PRINT FOREGROUND.



---

DR PRINT FOREGROUND (area; message; printDialog)

Parameter	Type		Description
area	Longint	→	4D Draw area
message	Integer	→	0 = Message off, 1 = Message on
printDialog	Integer	→	0 = Without dialog box, 1 = With dialog box

### Description

The command DR PRINT FOREGROUND is the reverse of the DR PRINT BACKGROUND command; it lets you print only objects that appear on the foreground.

- If message equals 1, a dialog box appears, allowing the user to cancel the printing in progress. If the user cancels the print job, DR Error returns error 55.
- If message equals 0, this dialog box does not appear, and the user cannot cancel the print job.

printDialog determines whether or not the standard Print File dialog box will appear.

- If printDialog equals 0, the standard Print File dialog box does not appear, and the print job begins immediately.
- If printDialog equals 1, the standard Print File dialog box appears.

### See Also

DR PRINT BACKGROUND.

---

DR PRINT MERGE (area; tableNum; message; printDialog)

Parameter	Type		Description
area	Longint	→	4D Draw area
tableNum	Integer	→	Table Number
message	Integer	→	0 = Message off, 1 = Message on
printDialog	Integer	→	0 = Without dialog, 1 = With dialog

### Description

The command DR PRINT MERGE allows you to print merge the current selection of tableNum. The document used for the print merge is specified by area.

If tableNum equals 0, the standard Print Merge dialog box is displayed.

- If message equals 1, a dialog box appears, allowing you to cancel the printing in progress. If you cancel the print job, DR Error returns an error 55.
- If message equals 0, this dialog box does not appear, and the user cannot cancel the print job.

printDialog determines whether the standard Print dialog box appears or not.

- If printDialog equals 0, the standard Print File dialog box does not appear and the print job begins immediately.
- If printDialog equals 1, the standard Print File dialog box appears.

### See Also

DR PRINT.

# 12

---

## DR References



---

DR INSERT EXPRESSION (area; scope; first; last; expression{; format})

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1 = All, 0 = Selected, >0 = Object ID
first	Integer	→	Position of first character minus 1
last	Integer	→	Position of last character
expression	String	→	Expression
format	String	→	Format of expression

### Description

The command DR INSERT EXPRESSION inserts a reference to expression into the text object in area described by scope.

- If scope equals -1, DR INSERT EXPRESSION inserts the reference into the first object of the document.
- If scope equals 0, the command inserts the reference into the first selected object.
- If scope is greater than 0, it must be equal to a specific text object's ID; the command inserts the reference within that text object. If the object does not exist, DR INSERT EXPRESSION does nothing, and DR Error returns error number 2.

If the object described by scope is not a text object, DR INSERT EXPRESSION does nothing, and DR Error returns error number 47.

first and last determine where the reference is inserted. first is one less than the first character position to be replaced and last is the last character position to be replaced. If first equals last, the command does not replace characters, and inserts the reference between first and first +1. If last is greater than the number of characters in the text object, DR INSERT EXPRESSION replaces characters from first to the end.

expression is the text equivalent of any valid 4th Dimension expression that returns a value. expression can be a reference to any of the following: a field, a variable, a 4th Dimension function, a user-defined function (project method), an external function, or a statement.

The following table gives examples for each expression type.

Example	Type
[Drawings]Object	Field
vCriteria	Variable
Current date	4th Dimension function
GetNum	User defined function (project method)
DR Count	4D Draw function
3 * "Hello"	Statement

The optional format parameter is the display format for the reference. This parameter is equivalent to choosing a format from the Format dialog box. Formats are referred to either by number or name; they are numbered in the order in which they appear in the list of the Format dialog box.

If format is a one or two digit string, then a format from the list is applied to expression. If format is not a one or two digit string, then it is compared against the text values of each format in the list. If it matches one of the values in the list, that format is applied. This means that you can refer to the first date format as either "19" or "Short".

If format is not in the list of formats, it is interpreted as a custom numeric format. If format is inappropriate for the resulting value of the reference, it is ignored. For instance, if you use a date format on a number, the number appears unformatted.

### Example

The following example creates a new text object, fills it with a reference to the 4th Dimension function Current date and formats it using the Long date format.

```
$ID := DR Draw text (Area;0.5;0.5;3.5;1;"Todays date is: ")  
⇒ DR INSERT EXPRESSION (Area;$ID;32000;32000;"Current date";"Long")
```

### See Also

DR INSERT FIELD.

---

DR INSERT FIELD (area; scope; first; last; table; field{; format})

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1 = All, 0 = Selected, >0 = Object ID
first	Integer	→	Position of first character minus 1
last	Integer	→	Position of last character
table	Integer	→	Table number of reference
field	Integer	→	Field number of reference
format	String	→	Format of reference

### Description

The command DR INSERT FIELD inserts a field reference into the text object in area described by scope.

- If scope equals -1, DR INSERT FIELD inserts the reference into the first object of the document.
  - If scope equals 0, the command inserts the reference into the first selected object.
  - If scope is greater than 0, it must be equal to a specific text object's ID, the command inserts the reference within that text object. If the object does not exist, DR INSERT FIELD does nothing, and DR Error returns error number 2.
- If the object described by scope is not a text object, DR INSERT FIELD does nothing, and DR Error returns error number 47.

first and last determine where the reference is inserted. first is one less than the first character position to be replaced and last is the last character position to be replaced. If first equals Last, no characters are replaced and the reference is inserted between first and first +1. If last is greater than the number of characters in the text object, DR INSERT FIELD replaces characters from first to the end in the text object.

table and field determine which field is referenced. table is the number of the table and field is the number of the field. Tables and fields are numbered in the order in which they were created.

The optional format parameter is the display format for the reference. This option is equivalent to choosing a format from the Format dialog box. Formats are referred to either by number or name; they are numbered in the order in which they appear in the list of the Format dialog box.

If format is a one or two digit string, then the format applied to field is from the list. If format is not a one or two digit string, it is compared against the text values of each format in the list. If it matches one of the values in the list, that format is applied. This means that you can refer to the first date format as either "19" or "Short".

If format is not in the list of formats, it is interpreted as a custom numeric format. If format is inappropriate for the resulting value of the reference, it is ignored. For instance, if you use a date format on a number, the number appears unformatted.

### Examples

(1) The following example inserts a reference to the first field of the first table into the text object that has an ID of 1, replacing any text in the object, and then formats it according to the eleventh format in the list.

⇒ *DR INSERT FIELD* (Area;1;0;32000;1;1;"11")

(2) You can use 4th Dimension's Field and Table functions to determine a field or table's number. This can make your code easier to read. For example, if the field used in the previous example is [Customers]Name, the code would look like this:

*DR INSERT FIELD* (Area;1;0;32000;Table(->[Customers]);Field(->Name);"11")

### See Also

DR INSERT EXPRESSION.



---

DR Place field (area; tableNum; fieldNum; format; location) → Longint

Parameter	Type		Description
area	Longint	→	4D Draw area
tableNum	Integer	→	Table Number
fieldNum	Integer	→	Field Number
format	String	→	Format to be used (not used if fieldNum is of type Picture)
location	Integer	→	0 = Normal, 1 = Centered, 2 = Origin
Function result	Longint	←	Object ID

### Description

The command DR Place field returns the ID number of the object created by inserting the fieldNum field from the tableNum Table. The object should be of type Picture if it is a Picture field and of type Text in all other cases. DR Place field works like the Insert Field menu item, except that it allows you to choose the location of the field inside area.

- If location equals 0, DR Place field places the field at the point of the last mouse click in the 4D Draw area.
- If location equals 1, DR Place field centers the field in the visible portion of the 4D Draw area.
- If location equals 2, DR Place field places the field at the origin, that is, at the point with coordinates (0,0).

For information on determining a table or field's number, see the example for DR INSERT FIELD.

### See Also

DR INSERT FIELD.

---

DR SET FORMAT (area; scope; format)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-3 = Selected characters, -1 = All, 0 = Selected >0 = Object ID
format	String	→	Format of reference

### Description

The command DR SET FORMAT makes format the display format for references in area described by scope.

- If scope equals -3, DR SET FORMAT formats any references in the highlighted text of the selected object.
- If scope equals -1, the command formats all references in the document.
- If scope equals 0, the command formats all references in the selected objects.
- If scope is greater than 0, it must be equal to a specific text object's ID; the command formats all references in that object. If the object does not exist, DR SET FORMAT does nothing, and DR Error returns error number 2.

If scope does not contain a reference, DR SET FORMAT does nothing.

format is the display format for the reference. It is the procedural equivalent of choosing a format from the format dialog box. Formats are referred to either by number or by name: they are numbered in the order in which they appear in the list of the Format dialog box.

If format is a one or two digit string, then the format applied is from the list. If format is not a one or two-digit string, then it is compared against the text values of each format in the list. If it matches one of the values in the list, that format is applied. This means that you can refer to the first date format as either "19" or "Short". If format is not in the list of formats it is interpreted as a custom numeric format. If format is inappropriate for the resulting value of the reference, it is ignored. For instance, if you use a date format on a number, the number appears unformatted.

### Example

The following example inserts the text "Current date", replacing the currently highlighted text of the selected text object. It then uses the DR DO COMMAND to turn it into a reference and applies the date format "Long".

```
DR SET TEXT (Area;-3;"Current date")
DR DO COMMAND (Area;7003)
⇒ DR SET FORMAT (Area;-3;"Long")
```

### See Also

DR SET TEXT.

# 13

---

## DR Rulers



---

DR ARRAY BASE TO SCALE (area; baseArray; scaleArray)

Parameter	Type	Description
area	Longint →	4D Draw area
baseArray	Numeric Array →	Array of base values
scaleArray	Numeric Array ←	Array of scale values

### Description

The command DR ARRAY BASE TO SCALE converts an array of base values into an array of scale values according to the ruler setup in area. Both arrays must be numeric (Integer, Longint, or Real); they can even be the same array.

After a call to DR ARRAY BASE TO SCALE, scaleArray contains the same number of elements as baseArray. DR ARRAY BASE TO SCALE is especially useful when working with the commands DR POLYGON TO ARRAY, DR Array to polygon, DR ATTRIBUTE TO ARRAY, and DR ARRAY TO ATTRIBUTE. When you use this command, you do not have to convert each array element individually.

### Example

The following example gets the coordinates of the vertices for the selected polygons and then converts them to scale units for display on screen. Note that in both calls to DR ARRAY BASE TO SCALE, the same array is used for baseArray and scaleArray.

```

    DR POLYGON TO ARRAY (Area;0;ArrayH;ArrayV)
⇒   DR ARRAY BASE TO SCALE (Area;ArrayH;ArrayH)
⇒   DR ARRAY BASE TO SCALE (Area;ArrayV;ArrayV)

```

### See Also

DR ARRAY SCALE TO BASE, DR GET RULER, DR SET RULER.

---

DR ARRAY SCALE TO BASE (area; scaleArray; baseArray)

Parameter	Type	Description
area	Longint →	4D Draw area
scaleArray	Numeric Array →	Array of scale values
baseArray	Numeric Array ←	Array of base values

### Description

The command DR ARRAY SCALE TO BASE converts an array of scale values into an array of base values according to the ruler setup in area. Both arrays must be numeric (Integer, Longint, or Real) and can be the same array. When you use this command, you do not have to convert each array element individually.

After a call to DR ARRAY SCALE TO BASE, baseArray contains the same number of elements as scaleArray. This command is especially useful when working with DR POLYGON TO ARRAY, DR Array to polygon, DR ATTRIBUTE TO ARRAY, and DR ARRAY TO ATTRIBUTE.

### Example

The following example uses the arrays from the example for DR ARRAY BASE TO SCALE. It converts the arrays back to base units and then creates a new polygon with the values.

```
⇒ DR ARRAY SCALE TO BASE (Area;ArrayH;ArrayH)
⇒ DR ARRAY SCALE TO BASE (Area;ArrayV;ArrayV)
   $ID := DR Array to polygon (Area;ArrayH;ArrayV)
```

### See Also

DR ARRAY BASE TO SCALE, DR GET RULER, DR SET RULER.

---

DR Base to scale (area; value) → Real

Parameter	Type		Description
area	Longint	→	4D Draw area
value	Real	→	Value in base units
Function result	Real	←	Value in scale units

### Description

The command DR Base to scale returns the size in scale units that value represents in area. value is specified in base units. One base unit is always equal to: Scale Value / Base Value scale units.

DR Base to scale is useful in conjunction with commands that return base values.

### Example

The following example gets the boundary of the selected objects and uses the values to display the width and height in scale units.

```

    DR GET BOUNDARY (Area;0;$Left;$Top;$Right;$Bottom)
⇒    vWidth := DR Base to scale (Area;$Right - $Left)
⇒    vHeight := DR Base to scale (Area;$Bottom - $Top)

```

### See Also

DR GET RULER, DR Scale to base, DR SET RULER.

DR GET ORIGIN (area; horizontal; vertical)

Parameter	Type		Description
area	Longint	→	4D Draw area
horizontal	Real	←	Horizontal position
vertical	Real	←	Vertical position

### Description

The command DR GET ORIGIN returns into the vertical and horizontal variables the origin for area.

horizontal and vertical are returned in base units. Use DR Base to scale to convert from base units to scale units. horizontal and vertical are returned as offsets from the upper left corner of the printable surface. Positive values indicate a position down or right. Negative values indicate a position up or left. The default position for the origin is the upper left corner of the printable surface of the document.

### Example

The following example is the object method for a button on a form. When the button is pressed, DR GET ORIGIN updates the process variables *vHorizontal* and *vVertical* to show the position of the origin.

⇒ *DR GET ORIGIN* (Area;vHorizontal;vVertical)

### See Also

DR SET ORIGIN.



---

DR GET RULER (area; baseU; scaleU; baseV; scaleV; divisions)

Parameter	Type		Description
area	Longint	→	4D Draw area
baseU	Integer	←	Base unit
scaleU	Integer	←	Scale unit
baseV	Real	←	Base value
scaleV	Real	←	Scale value
divisions	Integer	←	Number of minor divisions

### Description

The command DR GET RULER returns into the baseU, scaleU, baseV, scaleV, and divisions variables the setup for the ruler in area.

baseU describes the base units for area. The possible base units are inches (1), centimeters (6), or points (11). scaleU describes the scale units for area. scale units are displayed in the coordinates panel.

scaleU can be any of the possible unit types. baseV and scaleV describe the relationship between base units and scale units. One base unit corresponds to  $\text{scaleV} / \text{baseV}$  scale units. divisions describes the number of minor ruler divisions to display on screen.

The following table lists unit codes:

Code	Unit
1	Inches
2	Feet
3	Yards
4	Miles
5	Millimeters
6	Centimeters
7	Decimeters
8	Meters
9	Decameters
10	Kilometers
11	Points

## Example

The following example displays an alert that shows the current scale for Area.

```
ARRAY STRING(20;aUnit;11)
aUnit{1} := "Inches"
aUnit{2} := "Feet"
aUnit{3} := "Yards"
aUnit{4} := "Miles"
aUnit{5} := "Millimeters"
aUnit{6} := "Centimeters"
aUnit{7} := "Decimeters"
aUnit{8} := "Meters"
aUnit{9} := "Decameters"
aUnit{10} := "Kilometers"
aUnit{11} := "Points"
⇒ DR GET RULER (Area;$BaseU;$ScaleU;$BaseV;$ScaleV;$Divisions)
$Temp := "The current scale is:" + Char(13) + "1"
$Temp := $Temp+aUnit{$BaseU}+" = " +String ($ScaleV / $BaseV)+" "+aUnit{$ScaleU}
ALERT ($Temp)
```

## See Also

DR SET RULER.

---

DR GET RULER OPTIONS (area; rulerLines; notation; grid)

Parameter	Type		Description
area	Longint	→	4D Draw area
rulerLines	Integer	←	0 = Normal, 1 = Through objects
notation	Integer	←	0 = Decimal, 1 = English
grid	Real	←	Grid amount in scale units

### Description

The command DR GET RULER OPTIONS returns into the rulerLines, notation, and grid variables various information about the rulers in area.

rulerLines describes the treatment of ruler lines in area.

- If rulerLines equals 0, objects filled with a pattern other than none obscure the ruler lines.
- If rulerLines equals 1, the user can see the ruler lines through objects, regardless of the fill pattern.
- If rulerLines equals -1, no change is made to the notation.

notation describes the way values are displayed in the Coordinates panel.

- If notation equals 0, the coordinates are displayed in decimal notation.
- If notation equals 1, the coordinates are displayed in English notation (Fractions and Feet' Inches") when the base unit is inches.
- If notation equals -1, no change is made to the notation.

grid describes the spacing of the grid in area, expressed in scale units. Use the DR Scale to base function to convert from scale units to base units.

### Example

The following example divides the grid size in half, increasing the precision with which objects can be placed.

```
⇒ DR GET RULER OPTIONS (Area;$Lines;$Notation;$Grid)
   DR SET RULER OPTIONS (Area;-1;-1;$Grid / 2)
```

### See Also

DR SET RULER OPTIONS.

---

DR Scale to base (area; value) → Real

Parameter	Type		Description
area	Longint	→	4D Draw area
value	Real	→	Value in scale units
Function result	Real	←	Value in base units

### Description

The command **DR Scale to base** returns the size in base units that value represents in area. value is specified in scale units. The ratio of scale to base units is determined by the ruler setup. See the **DR GET RULER** command for more information. **DR Scale to base** is useful in conjunction with commands that use base values as parameters.

### Example

The following example uses the variables defined for the example in **DR Base to scale**. It takes the values of the variables, converts them back to base units and then uses them to size the selected object.

```
⇒ $Width := DR Scale to base (Area;vWidth)
⇒ $Height := DR Scale to base (Area;vHeight)
   DR SIZE (Area;0;0;2;2;$Width;$Height)
```

### See Also

**DR Base to scale**, **DR GET RULER**, **DR SET RULER**.

---

DR SET ORIGIN (area; horizontal; vertical; mode)

Parameter	Type		Description
area	Longint	→	4D Draw area
horizontal	Real	→	Horizontal position
vertical	Real	→	Vertical position
mode	Integer	→	0 = Absolute, 1 = Relative

### Description

The command DR SET ORIGIN sets the origin for area according to the horizontal, vertical, and mode parameters.

horizontal and vertical are specified in base units. Use DR Scale to base to convert from scale units to base units. You can specify horizontal and vertical as offsets from the current origin (relative) or as offsets from the upper left corner of the printable surface of the document (absolute).

- If mode equals 0, horizontal and vertical are absolute coordinates.
- If mode equals 1, horizontal and vertical are relative coordinates.

Positive values indicate a direction down or right. Negative values indicate a direction up or left. The default position for the origin is the upper left corner of the printable surface of the document. Many other 4D Draw commands rely on the position of the origin.

### Example

This example resets the origin to the upper left corner of the area.

⇒ *DR SET ORIGIN* (Area;0;0;0)

### See Also

DR GET ORIGIN.

---

DR SET RULER (area; baseU; scaleU; baseV; scaleV; divisions)

Parameter	Type		Description
area	Longint	→	4D Draw area
baseU	Integer	→	Base unit
scaleU	Integer	→	Scale unit
baseV	Real	→	Base value
scaleV	Real	→	Scale value
divisions	Integer	→	Number of minor divisions

### Description

The command DR SET RULER changes the basic ruler setup for area according to the baseU, divisions, baseV, scaleV, and scaleU parameters.

baseU specifies the base units for area. The possible base units are inches, centimeters, or points.

scaleU specifies the scale units for area. Scale units are displayed in the Coordinates panel. scaleU can be any of the possible unit types.

baseV and scaleV specify the relationship between base units and scale units. One base unit corresponds to scaleV / baseV scale units. Divisions specifies the number of minor ruler divisions to display on screen.

See the description of DR GET RULER for a complete list of unit codes.

### Example

The following example changes the ruler setup for Area to 1 inch equals 1 foot, with 8 minor ruler divisions.

⇒ *DR SET RULER* (Area;1;2;1;1;8)

### See Also

DR GET RULER.

---

DR SET RULER OPTIONS (area; rulerLines; notation; grid)

Parameter	Type		Description
area	Longint	→	4D Draw area
rulerLines	Integer	→	-1 = No change, 0 = Normal, 1 = Through objects
notation	Integer	→	-1 = No change, 0 = Decimal, 1 = English
grid	Real	→	Grid amount in scale units -1 = No change

### Description

The command DR SET RULER OPTIONS changes the ruler options for area according to the rulerLines, notation, and grid parameters.

rulerLines specifies the treatment of ruler lines in area.

- If rulerLines equals 0, objects filled with a pattern other than None obscure the ruler lines.
- If rulerLines equals 1, the user can see the ruler lines through objects, regardless of fill pattern.
- If rulerLines equals -1, no change is made to the ruler lines.

notation specifies the way values are displayed in the Coordinates panel.

- If notation equals 0, the coordinates are displayed in decimal notation.
- If notation equals 1, the coordinates are displayed in English notation (Fractions and Feet' Inches").
- If notation equals -1, no change is made to the notation.

grid describes the spacing of the grid in area, expressed in scale units. Use the DR Scale to base function to convert from scale units to base units. If grid equals -1, no change is made to the grid.

### See Also

DR GET RULER OPTIONS.





14

---

# DR Set Attributes



DR SET ARC SPECS (area; scope; start; length)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1 = All, 0 = Selected, >0 = Object ID
start	Integer	→	Starting angle in degrees
length	Integer	→	Length of arc in degrees

### Description

The command DR SET ARC SPECS reshapes the arc in area described by scope.

- If scope equals -1, the command affects the first object in the document.
- If scope equals 0, the command affects the first selected object.
- If scope is greater than 0, it must be equal to a specific arc's ID; DR SET ARC SPECS affects that arc. If the object does not exist, the command does nothing, and DR Error returns error number 2.

If the object described by scope is not an arc, DR SET ARC SPECS does nothing, and DR Error returns error number 47.

start is the beginning of the arc in degrees. This is the endpoint that begins the arc, moving in a counter-clockwise fashion.

length is the length of the arc in degrees. The end of an arc is  $\text{start} + \text{length} \% 360$ . Arcs have a maximum length of 359°.

See the diagram for the DR Draw arc function.

### Example

The following example reshapes the first ten objects in Area. All ten must be arcs and each is given a length of 18° with a different starting angle.

```

For ($i;0;9)
⇒  DR SET ARC SPECS (Area;$i + 1;$i * 18;18)
End for

```

### See Also

DR Draw arc, DR GET ARC SPECS, DR Get ID.

---

DR SET ATTRIBUTE LOCK (area; scope; attributeNum; lockState)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1 = All objects, 0 = Selected objects, >0 = ID
attributeNum	Integer	→	Number of the attribute to lock
lockState	Integer	→	0 = Unlock, 1 = Lock, 2 = Toggle

### Description

The command DR SET ATTRIBUTE LOCK locks or unlocks the selected attribute for the objects in scope.

The attributeNum parameter accepts one of the following codes:

Attributes	Values
<b>General</b>	
Name	3
Deletion	0
Location	7
Size	5
Shape	25
Rotation	11
Ungrouping	1
Corner Rounding	24
<b>Text</b>	
Font	19
Size	20
Style	21
Justification	22
Edition	23
<b>Line</b>	
Color	15
Pattern	14
Size	16
Endmarks	17
<b>Fill</b>	
Color	13
Pattern	12

---

DR SET CORNER ROUNDING (area; scope; amount)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2 = Default, -1 = All, 0 = Selected, >0 = ID
amount	Real	→	Corner rounding amount

### Description

The command DR SET CORNER ROUNDING changes the corner rounding for the objects in area described by scope. The corner rounding is specified in base units. Use the DR Scale to base function to convert from scale units to base units.

- If scope equals -2, the command sets the default corner rounding for new round rectangles.
- If scope equals -1, the command sets the corner rounding for all objects in the document.
- If scope equals 0, the command sets the corner rounding for the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID and that object's corner rounding is set. If the object does not exist, DR SET CORNER ROUNDING does nothing, and DR Error returns error number 2.

### See Also

DR Get corner rounding, DR Get ID.

DR SET ENDMARKS (area; scope; type; point)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2 = Default, -1 = All, 0 = Selected, >0 = ID
type	Integer	→	1 = Arrow, 2 = Bar
point	Integer	→	0 = None, 1 = Start, 2 = End, 3 = Both

### Description

The command DR SET ENDMARKS changes the endmarks for the lines in area described by scope.

- If scope equals -2, the command sets the default endmarks for new lines.
- If scope equals -1, the command sets the endmarks for all lines in the document.
- If scope equals 0, the command sets the endmarks for the selected lines.
- If scope is greater than 0, it must be equal to a specific line's ID and that line's endmarks are set. If the object does not exist, DR SET ENDMARKS does nothing, and DR Error returns error number 2.

type is the kind of endmark on the specified lines. Every line has a type of endmark, even though the endmarks may not be displayed.

- If type equals 1, the lines have arrow endmarks.
- If type equals 2, the lines have bar endmarks.

point is which ends of the line have endmarks.

- If point equals 0, the lines have no endmarks showing.
- If point equals 1, the lines have endmarks at the beginning.
- If point equals 2, the lines have endmarks at the ending.
- If point equals 3, the lines have endmarks on both ends.

### Example

The following example gives the selected lines an arrow endmark at their starting points.

⇒ *DR SET ENDMARKS* (Area;0;1;1)

### See Also

DR GET ENDMARKS.

---

DR SET FILL ATTRIBUTES (area; scope; pattern; color)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2 = Default, -1 = All, 0 = Selected, >0 = ID
pattern	Integer	→	Pattern index (-1 = No change)
color	Longint	→	Color value (-1 = No change)

### Description

The command DR SET FILL ATTRIBUTES changes the fill attributes for the objects in area described by scope. Fill attributes are determined by the interior of objects.

- If scope equals -2, the command sets the default fill attributes.
- If scope equals -1, the command sets the fill attributes for all objects in the document.
- If scope equals 0, the command sets the fill attributes for the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID and that object's attributes are set. If the object does not exist, DR SET FILL ATTRIBUTES does nothing, and DR Error returns error number 2.

pattern is the number of the pattern in the palette. Patterns have a value in the range of 1 through 36 and are numbered left to right, top to bottom.

color is a long integer that represents the color of the object's interior. This number can be obtained from the commands DR RGB to color and DR Index to color.

For each of the last two parameters (pattern and color), passing -1 will leave that attribute unchanged.

### Example

The following example sets the fill attributes for all objects in the document. It makes the object's patterns solid but doesn't change their color.

⇒ *DR SET FILL ATTRIBUTES* (Area;0;3;-1)

### See Also

DR GET FILL ATTRIBUTES.

DR SET HANDLE STATE (area; scope; action)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1 = All objects, 0 = Selected objects, >0 = ID
action	Integer	→	1 = Show, 0 = Hide

### Description

The command DR SET HANDLE STATE allows you to determine whether the selection handles for the objects in scope are visible or not. Selection handles are the black squares that appear around the object when it is selected, allowing you to resize the object.

If action equals 1, the object's handles will be visible. If action equals 0, the object's handles will be invisible.

The state of the handles is stored even after you quit the database.



---

DR SET HIGHLIGHT (area; scope; first; last)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1 = All, 0 = Selected, >0 = Object ID
first	Integer	→	Position of first character -1
last	Integer	→	Position of last character

### Description

The command DR SET HIGHLIGHT highlights characters within the text object in area described by scope.

- If scope equals -1, the command highlights characters in the first object of the document.
- If scope equals 0, the command highlights characters in the first selected object.
- If scope is greater than 0, it must be equal to a specific text object's ID and characters within that text object are highlighted. If the object does not exist, DR SET HIGHLIGHT does nothing, and DR Error returns error number 2. DR SET HIGHLIGHT causes the object described by scope to become the only object selected in area.

If the object described by scope is not a text object, DR SET HIGHLIGHT does nothing, and DR Error returns error number 47.

first and last determine which characters are highlighted. first is one less than the first character position to be highlighted. last is the last character position to be highlighted. If first equals last, no characters are selected and the insertion point is between first and first +1. If last is greater than the number of characters in the text object, then DR SET HIGHLIGHT highlights characters to the end of the text object.

DR SET HIGHLIGHT does not highlight only part of a reference. If any portion of a reference is highlighted, DR SET HIGHLIGHT adjusts the highlight to include the entire reference.

DR SET HIGHLIGHT is valid only when area is being viewed at 100 percent (actual size). If the drawing is displayed at a different magnification, DR SET HIGHLIGHT does nothing, and DR Error returns error number 48.

## Example

The following example gets the text of the selected text object and looks for the name "4th Dimension". If 4th Dimension is found, it highlights it and then makes it bold.

```
$Temp := DR Get text (Area;0)
$Find := Position ("4th Dimension";$Temp)
If ($Find # 0)
⇒   DR SET HIGHLIGHT (Area;0;$Find - 1;$Find + 12)
     DR SET TEXT ATTRIBUTES (Area;-3;-1;-1;1;-1;-1;-1)
End if
```

## See Also

DR Get text, DR GET TEXT ATTRIBUTES, DR SET TEXT, DR SET TEXT ATTRIBUTES.

---

DR SET LINE ATTRIBUTES (area{; area2; ...; areaN}; scope; pattern; color; width)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2 = Default, -1 = All, 0 = Selected, >0 = ID
pattern	Integer	→	Pattern index, -1 = No change
color	Longint	→	Color value, -1 = No change
width	Real	→	Line width in points, -1 = No change

### Description

The command DR SET LINE ATTRIBUTES changes the line attributes for the objects in area described by scope. Line attributes are determined by the border of objects.

- If scope equals -2, the command sets the default line attributes.
- If scope equals -1, the command sets the line attributes for all objects in the document.
- If scope equals 0, the command sets the line attributes for the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID and that object's attributes are set. If the object does not exist, DR SET LINE ATTRIBUTES does nothing, and DR Error returns error number 2.

pattern is the number of the pattern in the palette. Patterns have a value in the range of 1 through 36 and are numbered left to right, top to bottom.

color is a long integer that represents the color of the lines. This number can be obtained from the functions DR RGB to color and DR Index to color.

width is a real number that describes the width (thickness) of the line in points (1/72 of an inch).

Passing -1 for each of the last three parameters (pattern, color, and width) leaves that attribute unchanged.

### See Also

DR GET LINE ATTRIBUTES.

---

DR SET LINE SPECS (area; scope; startH; startV; endH; endV)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1 = All, 0 = Selected, >0 = Object ID
startH	Real	→	Horizontal position of start
startV	Real	→	Vertical position of start
endH	Real	→	Horizontal position of end
endV	Real	→	Vertical position of end

### Description

The command DR SET LINE SPECS resets the endpoints of the line in area described by scope.

- If scope equals -1, the command affects the endpoints for the first object in the document.
- If scope equals 0, the command affects the endpoints for the first selected object.
- If scope is greater than 0, it must be equal to a specific line's ID; the command affects that line's endpoints. If the object does not exist, DR SET LINE SPECS does nothing, and DR Error returns error number 2.

If the object described by scope is not a line, DR SET LINE SPECS does nothing, and DR Error returns error number 47.

The line is repositioned according to startH, startV, endH, and endV. All four coordinates are expressed in base units. Use the DR Scale to base function to convert from scale units to base units. startH, startV, endH, and endV are specified as offsets from the current origin.

Positive values indicate a position below or to the right of the origin. Negative values indicate a position above or to the left of the origin.

### Example

See the example for the DR GET LINE SPECS command.

### See Also

DR GET LINE SPECS, DR SET ENDMARKS, DR SET LINE ATTRIBUTES.

---

DR SET NAME (area; scope; name)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2 = Default, -1 = All, 0 = Selected, >0 = ID
name	String	→	Name

### Description

The command DR SET NAME makes name the name for the objects in the area described by scope.

A name is a string associated with an object and is not necessarily unique. Names have a maximum length of 31 characters. Names can be set by the user through the Object Attributes dialog box.

- If scope equals -2, the command sets the default name.
- If scope equals -1, the command sets the name for all objects in the document.
- If scope equals 0, the command sets the name for the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID; DR SET NAME and that object's name is set. If the object does not exist, DR SET NAME does nothing, and DR Error returns error code 2.

### Example

This example sets the names for all objects in area to empty strings.

⇒ *DR SET NAME* (Area;-1;"" )

### See Also

DR Get ID, DR GET NAME.

---

DR SET POLYGON VERTEX (area; scope; number; vertexH; vertexV)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1 = All, 0 = Selected, >0 = Object ID
number	Integer	→	Number of vertex to edit
vertexH	Real	→	Horizontal Position
vertexV	Real	→	Vertical Position

### Description

The command DR SET POLYGON VERTEX modifies the vertex specified by number for the polygon in area described by scope.

- If scope equals -1, the command affects the first object in the document.
- If scope equals 0, the command affects the first selected object.
- If scope is greater than 0, it must be equal to a specific polygon's ID and that polygon is affected. If the object does not exist, the command does nothing, and DR Error returns error number 2.

If the object described by scope is not a polygon, DR SET POLYGON VERTEX does nothing, and DR Error returns error number 47.

number is the number of the vertex within the polygon. Vertices are numbered in the order in which they were created. If number exceeds the number of vertices in the polygon, DR SET POLYGON VERTEX does nothing, and DR Error returns error number 49.

vertexH and vertexV are the new coordinates for the vertex. Both vertexH and vertexV are expressed in base units. Use the DR Scale to base function to convert from scale units to base units. vertexH and vertexV are specified as offsets from the current origin. Positive values indicate a position below or to the right of the origin. Negative values indicate a position above or to the left of the origin.

### Example

The following example gets the position of the first vertex of the polygon whose ID equals 1 and resets the vertex one unit down and one unit to the right.

```
⇒ DR GET POLYGON VERTEX (Area;1;1;$VertexH;$VertexV)
⇒ DR SET POLYGON VERTEX (Area;1;1;$VertexH + 1;$VertexV + 1)
```

### See Also

DR GET POLYGON VERTEX.

---

DR SET REFNUM (area; scope; refNum)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-2 = Default, -1 = All, 0 = Selected, >0 = ID
refNum	Longint	→	Reference number

### Description

The command DR SET REFNUM makes refNum the reference number for the objects in area described by scope.

A reference number is like an object name, in that it is a way to identify an object and it is not unique. A reference number is not an Object ID number, which is a unique number assigned by 4D Draw for each object in a document.

- If scope equals -2, the command sets the default reference number.
- If scope equals -1, the command sets the reference number for all objects in the document.
- If scope equals 0, the command sets the reference number for the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID; DR SET REFNUM sets that object's reference number. If the object does not exist, DR SET REFNUM does nothing, and DR Error returns error code 2.

A reference number is a non-unique long integer associated with an object. Reference numbers can be manipulated only procedurally. The default value for reference numbers is 0.

### Example

The following example changes the reference number of the selected objects to the value contained in the process variable *vNumber*.

⇒ *DR SET REFNUM* (Area;0;vNumber)

### See Also

DR Get ID, DR Get refnum.

---

DR SET TEXT (area; scope; characters)

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-3 = Selected characters, -1 = All, 0 = Selected objects, >0 = Object ID
characters	Text	→	Characters to set

### Description

The command DR SET TEXT fills the text objects in area described by scope with characters.

- If scope equals -3, the command replaces the highlighted text of the selected text object with characters. If the insertion point is between characters, DR SET TEXT inserts characters at the insertion point.
- If scope equals -1, the command replaces the text of the first object of the document.
- If scope equals 0, the command replaces the text in the first selected object.
- If scope is greater than 0, it must be equal to the ID of a specific text object; DR SET TEXT replaces that object's text. If the object does not exist, DR SET TEXT does nothing, and DR Error returns error number 2.

If the object described by scope is not a text object, DR SET TEXT does nothing, and DR Error returns error number 47.

### Example

The following example makes the characters in the process variable *vText* the text for the selected text objects.

⇒ *DR SET TEXT* (Area;0;vText)

### See Also

DR Get text.



---

DR SET TEXT ATTRIBUTES (area; scope; font; size; style; just{; frame{; expansion}})

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-3 = Selected characters, -2 = Default, -1 = All, 0 = Selected objects, >0 = Object ID
font	Integer	→	Font number (-1 = No change)
size	Integer	→	Font size in points (-1 = No change)
style	Integer	→	Font style (-1 = No change)
just	Integer	→	Font justification (-1 = No change)
frame	Integer	→	-1 = No change, 0 = Variable, 1 = Fixed
expansion	Integer	→	-1 = No change, 0 = Down, 1 = Up

### Description

The command DR SET TEXT ATTRIBUTES changes the text attributes for the text or objects in area described by scope.

- If scope equals -3, the command affects the text attributes for the highlighted text of the selected object.
- If scope equals -2, the command sets the default text attributes.
- If scope equals -1, the command sets the text attributes for all objects in the document.
- If scope equals 0, the command sets the text attributes for the selected objects.
- If scope is greater than 0, it must be equal to the ID of a specific text object; DR SET TEXT ATTRIBUTES sets that object's attributes. If the object does not exist, DR SET TEXT ATTRIBUTES does nothing, and DR Error returns error code 2.

The following table summarizes the possible values for scope:

Scope	Affected text/objects
-3	Highlighted text of selected object
-2	Default text attributes
-1	Text attributes for all objects
0	Text attributes for selected objects
>0	Text attributes for a specific object

font is the ID of the font in your system. You can determine the font ID by passing the font name to the DR Font number function.

size is the size in points of the highlighted text or text objects.

style is a number that is obtained by adding several style numbers. Style numbers are presented in the following list:

Value	Style
0	Plain
1	Bold
2	Italic
4	Underline
8	Outlined
16	Shadowed

just is the alignment of the text within the text block. The following table summarizes the possible values for just:

Value	Justification
0	Left
1	Center
2	Right

The optional frame parameter controls whether the text object has a fixed size. If frame equals 0 or is not specified, the height of the object is variable and is determined by the amount of text and the width of the text object. If frame equals 1, the height of the object is fixed and any text that does not fit within the text object is truncated. If frame is omitted, flow must also be omitted.

expansion describes which direction text will flow within the object when it wraps to the next line.

- If expansion equals 0, the text object expands down the page to accommodate a new line of text.
- If expansion equals 1, the text object expands up the page when a line of text is added.

frame and expansion are especially useful when working with text objects that contain references. As references can insert any amount of text, having control over the size and flow direction of text objects during printing can be useful.

Passing -1 for each of the last six parameters (font, size, style, just, frame, and expansion) will leave that attribute unchanged.

### Example

The following example sets the text attributes for all text objects in the document. It leaves the same font but makes all text 12-point, plain, and left justified.

⇒ *DR SET TEXT ATTRIBUTES* (Area;-1;-1;12;0;0;-1;-1)

### See Also

DR GET TEXT ATTRIBUTES.

# 15

---

## DR Utilities



---

DR Calculate area (area; scope) → Real

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1 = All, 0 = Selected, >0 = Object ID
Function result	Real	←	Surface area

### Description

The command DR Calculate area returns the surface area for the objects in area described by scope.

- If scope equals -1, DR Calculate area returns the surface area of all objects in the document.
- If scope equals 0, DR Calculate area returns the surface area of the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID; DR Calculate area returns the surface area of the object.

If the object does not exist, DR Calculate area returns -3, and DR Error returns error code 2.

### Example

The following example calculates the surface area of all objects in area.

```
⇒ $SurfArea := DR Calculate area (Area;-1)
```

### See Also

DR Calculate perimeter, DR Get ID.

---

DR Calculate perimeter (area; scope) → Real

Parameter	Type		Description
area	Longint	→	4D Draw area
scope	Longint	→	-1 = All, 0 = Selected, >0 = Object ID
Function result	Real	←	Perimeter of the objects described by scope

### Description

The command DR Calculate perimeter returns the perimeter of the objects described by scope.

- If scope equals -1, DR Calculate perimeter returns the perimeter of all objects in the document.
- If scope equals 0, DR Calculate perimeter returns the perimeter of the selected objects.
- If scope is greater than 0, it must be equal to a specific object's ID; DR Calculate perimeter returns the perimeter of the object.

If the object does not exist, DR Calculate perimeter returns -32000 and DR Error returns error number 2.

### Example

The following example calculates the perimeter of the selected objects in area.

```
⇒ $Perimeter := DR Calculate perimeter (Area;0)
```

### See Also

DR Calculate area, DR Get ID.

---

DR Clipboard to picture → Picture

Parameter	Type	Description
-----------	------	-------------

This command does not require any parameters

Function result	Picture	← 4th Dimension picture
-----------------	---------	-------------------------

### Description

The command DR Clipboard to picture returns a 4th Dimension picture that is a copy of the contents of the Clipboard. The picture can then be used as a standard 4th Dimension picture value. If the Clipboard does not contain a picture, DR Clipboard to picture returns an empty picture, and DR Error returns error code 14.

### Example

The following example places the contents of the Clipboard into a Picture field.

⇒ [Drawing]Object := *DR Clipboard to picture*

### See Also

DR PICTURE TO CLIPBOARD.

---

DR Color to index (color) → Integer

Parameter	Type		Description
color	Longint	→	Color value
Function result	Integer	←	Index of color on the 4D color palette

### Description

The command DR Color to index returns the index of the color closest to color on the 4th Dimension color palette.

For example, if a particular shade of blue is specified by color, DR Color to index returns the closest shade of blue found on the 4th Dimension palette. Colors on the 4th Dimension palette are numbered 1 through 256.

### Example

The following example places into the *Color* variable the palette index of the color specified by the RGB values:

```
⇒ Color := DR Color to index (DR RGB to color (65535;0;18405))
```

### See Also

DR Index to color.



---

DR COLOR TO RGB (color; red; green; blue)

Parameter	Type		Description
color	Integer	→	Color
red	Longint	←	Red value
green	Longint	←	Green value
blue	Longint	←	Blue value

### Description

The command DR COLOR TO RGB maps the color defined by color into its three components: red, green, and blue.

color is an internal number used by 4D Draw and can be obtained through the functions DR RGB to color, DR Index to color, DR GET FILL ATTRIBUTES, and DR GET LINE ATTRIBUTES.

On Mac OS, the obtained values are the same as the System Chromatic Wheel (Color Picker). On Windows, they are the same as the custom color selector values multiplied by 256.

### Example

The following example gets the fill color of the selected objects and then breaks it down into its three components.

```

    DR GET FILL ATTRIBUTES (Area;0;$Pattern;$Color)
⇒   DR COLOR TO RGB ($Color;$Red;$Green;$Blue)

```

### See Also

DR RGB to color.

DR Font name (fontNumber) → String

Parameter	Type		Description
fontNumber	Integer	→	Font ID number
Function result	String	←	Font name

### Description

The command DR Font name returns the name of the font whose ID is fontNumber. The ID is the value returned by DR Font number.

If fontNumber does not exist, DR Font name returns an empty string.

### Example

The following example places into the *vName* variable the font whose ID is 3.

⇒ `vName := DR Font name (3)`

### See Also

DR Font number.

DR Font number (fontName) → Integer

Parameter	Type		Description
fontName	Text	→	Name of font
Function result	Integer	←	Font ID

### Description

The command DR Font number returns the integer ID for the font named fontName. The ID is used in the DR SET TEXT ATTRIBUTES command.

### Example

The following example changes the font of the currently selected objects in Area to Times.

```
⇒ $FontNum:=DR Font number ("Times")  
   DR SET TEXT ATTRIBUTES (Area;0;$FontNum;-1;-1;-1;-1;-1)
```

### See Also

DR Font name.

---

DR Index to color (index) → Longint

Parameter	Type		Description
index	Integer	→	Palette index
Function result	Longint	←	Color described by the palette index

### Description

The command DR Index to color is a convenient way to specify a color without knowing the individual color components, Red, Green, and Blue. Colors on the 4th Dimension palette are numbered 1 through 256. DR Index to color returns the color described by the palette index index.

index is an integer that specifies a particular color on the 4th Dimension palette.

### Example

The following example sets the color of the currently selected objects in Area to light blue (Cyan).

⇒ *DR SET FILL ATTRIBUTES (Area;0;3;DR Index to color (8))*

### See Also

DR Color to index.

DR PICTURE TO CLIPBOARD (picture)

Parameter	Type		Description
picture	Picture	→	4th Dimension Picture

### Description

The command DR PICTURE TO CLIPBOARD copies picture to the Clipboard. Once on the Clipboard, picture can be pasted anywhere that pictures are allowed.

### Example

The following example copies the contents of the Picture field [Drawing]Object to the Clipboard.

⇒ *DR PICTURE TO CLIPBOARD* ([Drawing]Object)

### See Also

DR Clipboard to picture.

DR RGB to color (red; green; blue) → Longint

Parameter	Type		Description
red	Longint	→	Red component MacOS: 0 to 65535 Windows: (0 to 255)*256
green	Longint	→	Green component MacOS: 0 to 65535 Windows: (0 to 255)*256
blue	Longint	→	Blue component MacOS: 0 to 65535 Windows: (0 to 255)*256
Function result	Longint	←	Color number in compact form

### Description

The command DR RGB to color returns a number in compact form, which is used by 4D Draw to manage colors. This number represents the three component colors, red, green, and blue. The red, green, and blue parameters are the same values used in your system's color picker.

The following table shows the values for red, green, and blue in commonly used colors.

Color	Macintosh			Windows		
	Red	Green	Blue	Red	Green	Blue
Red	56576	2048	1536	221	8	6
Green	0	32768	4352	0	128	17
Blue	0	0	54272	0	0	212
Cyan	512	43776	59904	2	171	234
Magenta	64512	62208	1280	252	243	5
Yellow	61952	2048	33792	242	8	132

### Example

The following example sets the selected objects in Area to Red.

```
⇒ $red := DR RGB to color (56683;2242;1698)
   DR SET FILL ATTRIBUTES (Area;0;3;$color)
```

### See Also

DR COLOR TO RGB.

16

---

# Appendixes





Attributes are the characteristics of objects, such as fill pattern or color. In 4D Draw several commands refer to attributes using numeric codes. Use this Appendix to determine the correct field types when binding attributes to fields, and the correct array types when filling arrays and modifying objects in the commands DR ATTRIBUTE TO ARRAY and DR ARRAY TO ATTRIBUTE.

The Ideal type is the best possible choice. Types listed under Other types are alternates but some data may be lost when information is transferred. Note the restrictions on certain attributes.

Code	Attribute	Ideal Type	Other Types	Restrictions
0	ID	Longint	Integer, Real	Can't be modified or searched
1	Object Type	Integer	Longint, Real	Can't be modified
2	Reference Number	Longint	Integer, Real	
3	Name	Alpha/String 31	Text	
4	Lock Code	Longint	Integer, Real	Can't be searched
5	Width	Real	Integer, Longint	Can't be searched
6	Height	Real	Integer, Longint	Can't be searched
7	Left Boundary	Real	Integer, Longint	Can't be searched
8	Right Boundary	Real	Integer, Longint	Can't be searched
9	Top Boundary	Real	Integer, Longint	Can't be searched
10	Bottom Boundary	Real	Integer, Longint	Can't be searched
11	Rotation	Integer	Longint, Real	
12	Fill Pattern	Integer	Longint, Real	
13	Fill Color	Longint	Integer, Real	
14	Line Pattern	Integer	Longint, Real	

Code	Attribute	Ideal Type	Other Types	Restrictions
15	Line Color	Longint	Integer, Real	
16	Line Width	Real	Integer, Longint	
17	Endmark Kind	Integer	Longint, Real	Can't be searched
18	Endmark End	Integer	Longint, Real	Can't be searched
19	Text Font	Integer	Longint, Real	
20	Text Size	Integer	Longint, Real	
21	Text Style	Integer	Longint, Real	
22	Text Just	Integer	Longint, Real	
23	Text Characters	Text	Alpha/String 255	Can't be searched
24	Corner Rounding	Real	Integer, Longint	
25	Area	Real	Integer, Longint	Can't be modified or searched
26	Perimeter/Line Length	Real	Integer, Longint	Can't be modified or searched
27	Objects	Integer	Integer	0 = Invisible, 1 = Visible
28	Objects	Integer	Integer	0 = Not in background 1 = In background

---

The following table lists the codes for 4D Draw error messages:

Error	Message
1	Invalid 4D Draw area reference.
2	Invalid object ID.
3	Invalid font.
4	Invalid select action.
5	Invalid object index.
6	Invalid command number.
7	Invalid value in DR SELECT BY ATTRIBUTE.
8	Insufficient memory.
9	Invalid alignment code.
10	Invalid display code.
11	Invalid ungroup level.
12	Invalid color index.
13	Invalid RGB value.
14	The Clipboard does not contain a picture.
15	Invalid degree amount.
16	Default cannot be set for this attribute.
17	Invalid array type.
18	Invalid array size.
19	Invalid pattern index.
20	Invalid line width.
21	Invalid corner rounding amount.
22	Invalid attribute code.
23	Hot link was not found.
24	Invalid file or field number.
25	Invalid endmark type.
26	Invalid endmark end.

Error	Message
27	Invalid text justification.
28	Object(s) specified do not have this attribute.
29	Muiltiple values.
30	Picture was not created.
31	Invalid field type.
32	Invalid Scope in DR AREA TO AREA.
33	Objects were not copied.
34	Last point clicked has not been set.
35	DR Draw text requires at least one character.
36	Source and destination areas must be different.
37	There are no selected objects.
38	All of these objects have this attribute locked.
39	Could not get picture.
40	Could not set picture.
41	Invalid object boundary.
42	Invalid Bind ID.
43	This bind is currently in use.
44	This attribute already exists in this bind.
45	A hot link with that name and type already exists.
46	There are no objects in this document.
47	This operation cannot be performed on this type of object.
48	Text can only be edited at actual size.
49	Invalid polygon vertex number.
50	Polygons must have at least 3 vertices.
51	Not enough memory to complete operation.
52	Invalid file type.
53	The version of this 4D Draw document is no longer supported.
54	This 4D Draw document was created with a later version.
55	User cancelled dialog.
56	Not in text editing mode.
57	Invalid scope.
58	One or more values are out of range.
59	Not enough objects to create a group.

Error	Message
60	Invalid character positions.
61	Cannot edit rotated or flipped text.
62	Invalid coordinate(s).
63	Invalid document size.
64	Invalid object size.
65	Invalid hot link type.
66	Adding to this hot link would generate recursion in the hot link chain.
67	Invalid color value.
68	This operation would cause objects to be moved outside the document.
69	This command is disabled.
70	Invalid scaling values.
71	This object has editing locked.
72	This operation would cause the maximum number of objects to be exceeded.
73	All groups are locked.
74	There are no foreground objects.
75	There are no background objects.
76	Bitmapped image is too large.
77	There are no visible objects.
78	There are no invisible objects.
79	There are no files.
80	Zoom percentage is out of range.
81	Pathname exceeds 255 characters.
82	This polygon needs a starting point.
32767	Miscellaneous error.
<0	Macintosh System error.

---

The following table lists the codes for the Event parameter.

Value	Event
-1	All Events
0	No Events
1	Area creation
2	Area deletion
4	Area activated (clicked or brought to the front)
8	Area deactivated (area is no longer active)
16	Object creation (create, paste, duplicate)
32	Object deletion (delete, cut, clear)
64	Command-Click (Macintosh) or Ctrl-Click (Windows), not necessarily on object
128	Object Moved (nudge, alignment, move, etc.)
256	Object Resized (arrow keys, drag, etc.)
512	Object Rotated
1024	Change in selected object(s)
2043	Double-click
4096	Reshaping of an object This event occurs when you reshape a polygon, line or arc; smooth or unsmooth it; or add or delete a polygon vertex.

The following table lists the 4D Draw object codes.

<b>Code</b>	<b>Description</b>
1	Text object
3	Image
4	Bitmap
5	Rectangle/Rounded Rectangle
6	Polygon/Freehand
7	Oval
8	Arc
9	Line
10	Group

The following table lists the ruler unit codes used by the commands DR SET RULER and DR GET RULER.

<b>Code</b>	<b>Description</b>
1	Inches
2	Feet
3	Yards
4	Miles
5	Millimeters
6	Centimeters
7	Decimeters
8	Meters
9	Decameters
10	Kilometers
11	Points



The following are the codes for the command parameter, used for menu item commands:

Menu	Command	Code
File	New...	1001
	Open...	1002
	Save	1003
	Save as...	1004
	Save as Template	1006
	Page Setup	1008
	Print	1009
	Go to Full Page	1011
	Print Merge	1012
	Import...	1013
	Export Selection as...	1014
Edit	Undo	2001
	Cut	2003
	Copy	2004
	Paste	2005
	Clear	2006
	Duplicate	2007
	Select All	2009
Text	Font menu	3001
	Individual font names	8001-8999
	Size menu	3002
	Individual font sizes	9001-9999
	Style menu	3003
	Plain	10001
	Bold	10002
	Italic	10003
	Underline	10004
	Outline	10005
	Shadow	10006
	Justification menu	3004
	Left	11001
	Center	11002
	Right	11003
Specifications...	3005	

Object	Fill Pattern menu	4001
	Individual fill patterns	12001-12036
	Fill Color menu	4002
	Individual fill colors	13001-13256
	Line Pattern menu	4004
	Individual line patterns	14001-14256
	Line Color menu	4005
	Individual line colors	15001-15256
	Line Width menu	4007
	Hairline	16001
	1 pt.	16002
	2 pts.	16003
	4 pts.	16004
	6 pts.	16005
	Other	16006
	Endmarks menu	4008
	No Endmarks	17001
	Start	17002
	End	17003
	Both Ends	17004
	Arrow endmark	17006
	Bar endmark	17007
	Smooth	4010
	Unsmooth	4011
	Reshape	4012
	Rotate	4013
	Round corners...	4015
	Lock	4017
	Attributes	4018
	Hide	4020
	Show All	4021
Arrange	Bring to Front	5001
	Send to Back	5002
	Move Forward	5003
	Move Backward	5004
	Align Objects	5006
	Align to Grid	5007
	Flip Horizontal	5009
	Flip Vertical	5010
	Group	5011
	Ungroup	5012
Add to Background	5014	

Layout	Actual Size	6001
	Fit to Window	6002
	Reduce	6003
	Enlarge	6004
	Display menu	6006
	Rulers	18001
	Ruler Lines	18002
	Page Breaks	18003
	Coordinates	18004
	Menu Bar	18005
	Tool Palette	18006
	Scroll Bars	18007
	Snap to Grid	6008
	Drawing Size	6010
	Set Up Rulers	6011
	Set Preferences	6012
	Release Background	6014
Database	Paste Field	7001
	Format	7002
	Reference	7003
	Unreference	7004
	Show Values/Show References	7006
	Binding	7014

Codes used by the 4D Draw Tool Palette:

1 Pointer



2 Text



3 Line



4 Rectangle



5 Round rectangle



6 Oval/Circle



7 Arc



8 Polygon



9 Freehand



# Command Index

## A

DR ACTIVATE BIND.....	93
DR ADD TO BACKGROUND.....	27
DR ADD TO BIND.....	94
DR ADD TO BITMAP.....	157
DR ALIGN.....	158
DR AREA TO AREA.....	73
DR AREA TO FIELD.....	74
DR Area to picture.....	76
DR ARRAY BASE TO SCALE.....	197
DR ARRAY SCALE TO BASE.....	198
DR ARRAY TO ATTRIBUTE.....	131
DR Array to polygon.....	133
DR ATTRIBUTE TO ARRAY.....	135

## B

DR Base to scale.....	199
-----------------------	-----

## C

DR Calculate area.....	229
DR Calculate perimeter.....	230
DR Clipboard to picture.....	231
DR Color to index.....	232
DR COLOR TO RGB.....	233
DR COORDINATES.....	55
DR Count.....	161

## D

DR DEACTIVATE BIND.....	95
DR DELETE.....	162
DR DELETE BIND.....	96
DR DELETE OFFSCREEN AREA.....	77

DR DISPLAY OPTIONS.....	56
DR DO COMMAND.....	28
DR Draw arc.....	141
DR Draw line.....	143
DR Draw oval.....	144
DR Draw rectangle.....	145
DR Draw text.....	147

## E

DR End polygon.....	148
DR Error.....	29
DR EVENT FILTER.....	30
DR EXPERT COMMAND.....	31
DR EXPERT MODE.....	32

## F

DR FIELD TO AREA.....	78
DR Font name.....	234
DR Font number.....	235

## G

DR GET ARC SPECS.....	101
DR GET AREA BOUNDARY.....	33
DR Get attribute lock.....	103
DR GET BOUNDARY.....	104
DR Get corner rounding.....	105
DR Get display.....	57
DR GET DOCUMENT SIZE.....	58
DR Get draw mode.....	59
DR GET ENDMARKS.....	106
DR GET FILL ATTRIBUTES.....	108
DR GET GLOBAL PREFERENCES.....	60
DR Get handle state.....	110
DR GET HIGHLIGHT.....	111

DR Get ID.....	112
DR GET LINE ATTRIBUTES.....	114
DR GET LINE SPECS.....	115
DR GET MOUSE.....	36
DR Get name.....	117
DR Get object type.....	119
DR GET ORIGIN.....	200
DR GET POLYGON VERTEX.....	121
DR GET PREFERENCES.....	61
DR Get refnum.....	122
DR Get rotation.....	123
DR GET RULER.....	201
DR GET RULER OPTIONS.....	203
DR Get text.....	124
DR GET TEXT ATTRIBUTES.....	125
DR Get text width.....	127
DR Get update mode.....	35
DR Get zoom.....	37
DR GROUP.....	163

## H

DR HIDE.....	164
--------------	-----

## I

DR Index to color.....	236
DR INSERT EXPRESSION.....	189
DR INSERT FIELD.....	191

## L

DR LAST CLICK.....	38
DR Last event.....	39
DR LOCK.....	165

## M

DR MENU STATUS.....	40
DR MOVE.....	167

## N

DR New bind.....	97
DR NEW DRAWING.....	79
DR New offscreen area.....	80

## O

DR Objects to bitmap.....	149
DR ON ERROR.....	41
DR ON EVENT.....	42
DR ON MENU.....	44
DR OPEN DOCUMENT.....	81

## P

DR PICTURE TO AREA.....	83
DR PICTURE TO CLIPBOARD.....	237
DR Place field.....	193
DR PLACE PICTURE.....	150
DR POLYGON CURVE.....	151
DR POLYGON LINE.....	153
DR POLYGON TO ARRAY.....	137
DR PRINT.....	183
DR PRINT BACKGROUND.....	184
DR PRINT FOREGROUND.....	185
DR PRINT MERGE.....	186



## R

DR REDRAW.....	45
DR RELEASE BACKGROUND.....	46
DR REMOVE FROM BACKGROUND.....	47
DR REMOVE FROM BIND.....	98
DR RGB to color.....	238
DR ROTATE.....	168

## S

DR SAVE DOCUMENT.....	84
DR SCALE.....	169
DR Scale to base.....	204
DR SCROLL DOCUMENT.....	48
DR SELECT.....	177
DR SELECT BY ATTRIBUTE.....	178
DR SELECT BY REGION.....	180
DR SET ARC SPECS.....	211
DR SET ATTRIBUTE LOCK.....	212
DR SET CORNER ROUNDING.....	213
DR SET DISPLAY.....	63
DR SET DOCUMENT SIZE.....	65
DR SET DRAW MODE.....	66
DR SET ENDMARKS.....	214
DR SET ENTERABLE.....	49
DR SET FILL ATTRIBUTES.....	215
DR SET FORMAT.....	194
DR SET GLOBAL PREFERENCES.....	67
DR SET HANDLE STATE.....	216
DR SET HIGHLIGHT.....	217
DR SET LINE ATTRIBUTES.....	219
DR SET LINE SPECS.....	220
DR SET NAME.....	221
DR SET ORIGIN.....	205
DR SET POLYGON VERTEX.....	222
DR SET PREFERENCES.....	69
DR SET REFNUM.....	223

DR SET RULER.....	206
DR SET RULER OPTIONS.....	207
DR SET TEXT.....	224
DR SET TEXT ATTRIBUTES.....	225
DR SET UPDATE MODE.....	50
DR SIZE.....	171
DR START POLYGON.....	154

## U

DR UNGROUP.....	173
-----------------	-----

## Z

DR ZOOM.....	51
--------------	----